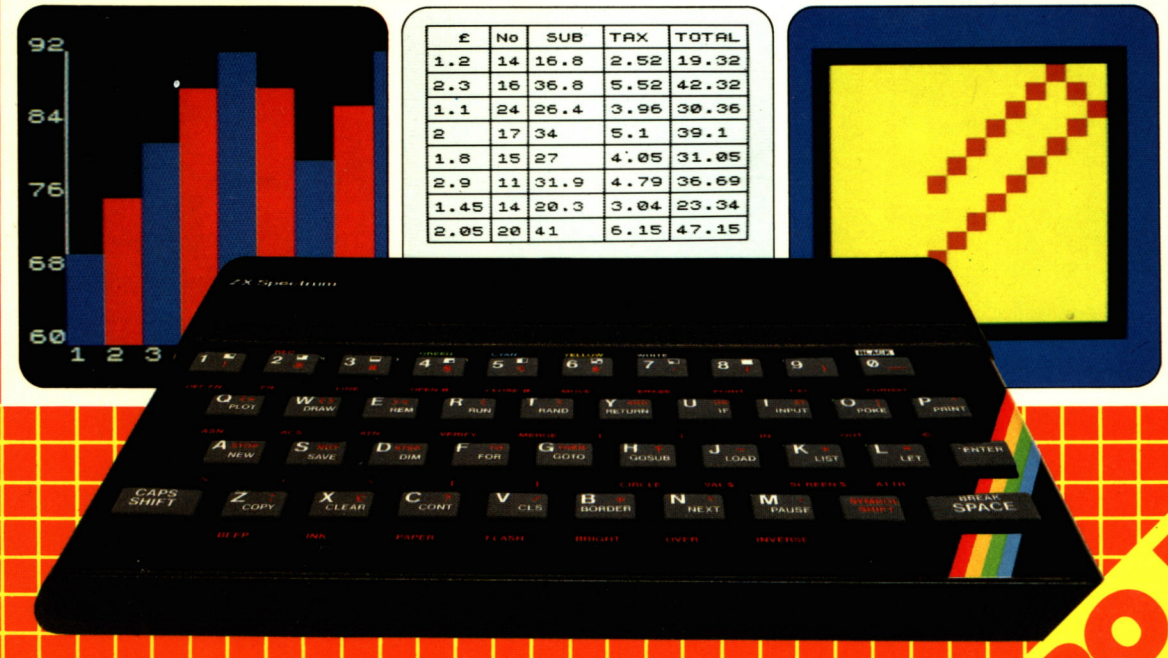
MicroTextos

Curso Visual en Pantalla

PROGRAMACION PASO-A-PASO

ZX SPECTRUM



IAN GRAHAM

LIBRO DOS

Técnicas adicionales para apasionantes juegos y gráficos a todo color.

MicroTextos

Curso Visual en Pantalla

PROGRAMACION PASO-A-PASO

ZX SPECTRUM

LA SERIE «CURSO VISUAL EN PANTALLA» SOBRE PROGRAMACION

Hoy día hay una urgente necesidad de guías prácticas, bien hechas y sencillas, para aprender a usar un ordenador. Por eso se han creado las Series sobre Programación «Curso Visual en Pantalla». Son un concepto totalmente nuevo en el campo de la autoenseñanza de microinformática y constituyen la primera biblioteca de manuales de Programación Paso-a-Paso, altamente ilustrados, para máquinas específicas.

LIBROS SOBRE EL ZX SPECTRUM

Este es el Libro Dos de la excepcional serie de guías «Paso-a-Paso» para programar el ZX Spectrum. Junto con los otros volúmenes, constituye un curso de autoenseñanza que comienza con los principios básicos de programación y llega a alcanzar, a través de técnicas y rutinas más complejas, un nivel avanzado.

LIBROS SOBRE OTROS ORDENADORES

Los otros volúmenes de la serie abarcarán los ordenadores más populares en el mundo. Estos incluirán:

Programación Paso-a-Paso para el **ZX Spectrum+**

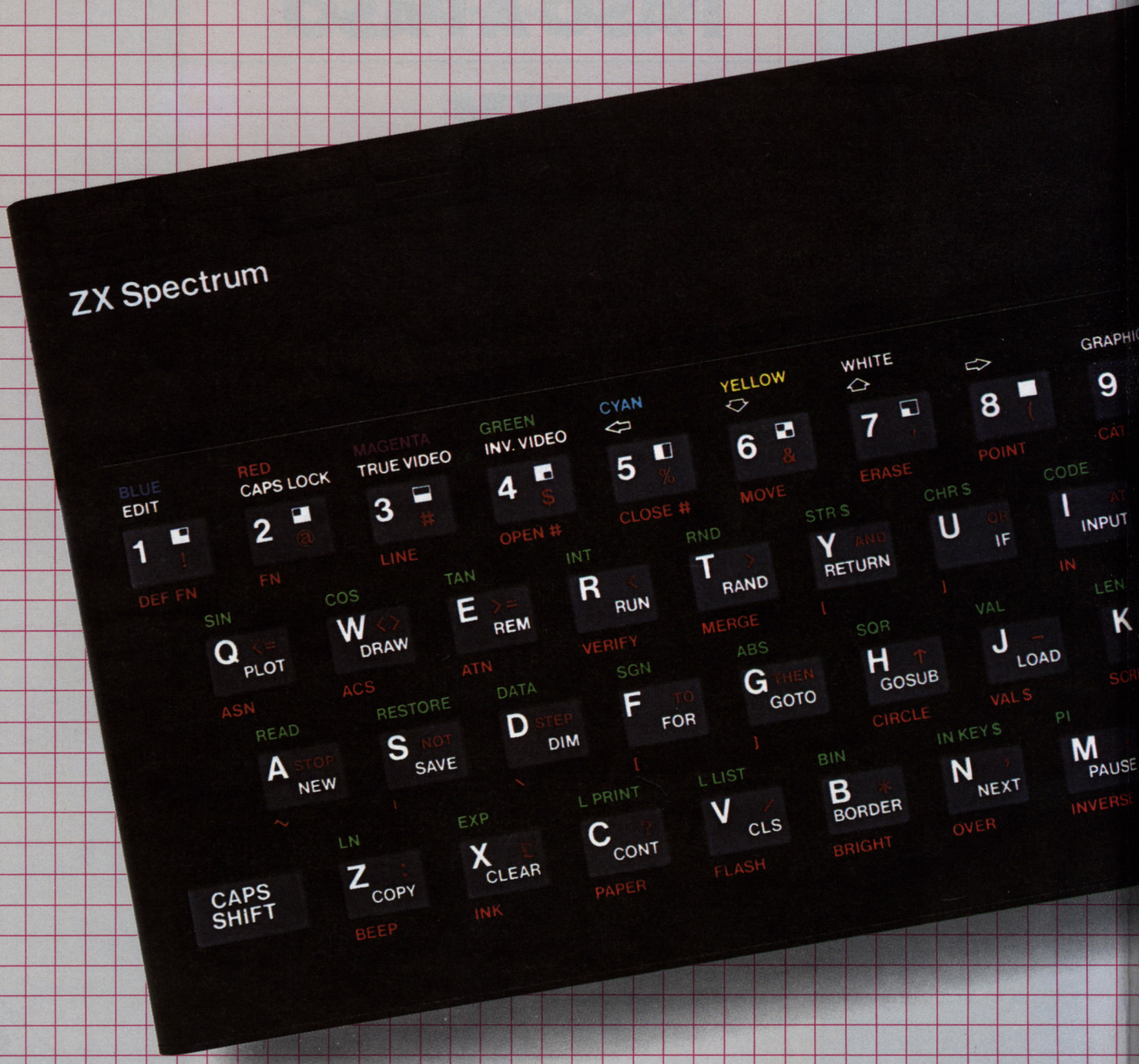
Programación Paso-a-Paso para el **Apple IIe**

Programación Paso-a-Paso para el **Commodore 64**

IAN GRAHAM

Después de licenciarse en Física Aplicada y obtener un diploma de Periodismo para graduados en la City University, Londres, Ian Graham ha trabajado como editor auxiliar de Electronics Today International y editor suplente de Which Video? Desde 1982, en que llegó a ser un escritor independiente de plena dedicación, ha colaborado en una amplia gama de revistas técnicas incluyendo Computing Today, Video Today, Video Search, Hobby Electronics, Electronic Insight, Popular HI-FI, Science Now, Next y otras, y también ha escrito varios libros muy conocidos sobre ordenadores e informática. Entre estos están Computer & Video Games, Information Technology, The inside Story-Computers.

LIBRO DOS

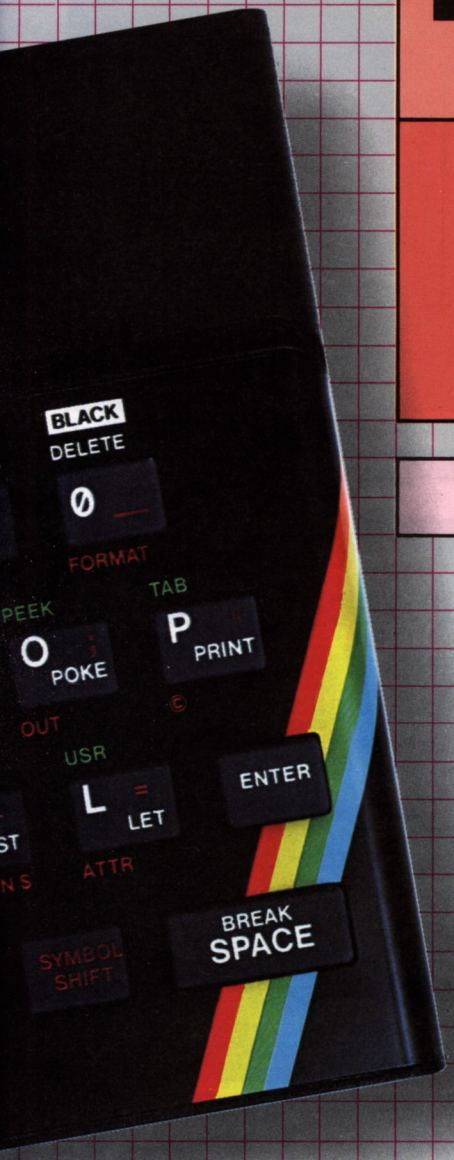

ZX Spectrum
BLUE EDIT 1 DEF FN
RED CAPS LOCK 2 FN
MAGENTA TRUE VIDEO 3 LINE
GREEN INV. VIDEO 4 OPEN #
CYAN 5 CLOSE #
YELLOW 6 MOVE
WHITE 7 ERASE
8 POINT
9
Q PLOT
W DRAW
E REM
R RUN
T RAND
Y RETURN
U IF
I INPUT
A NEW
S SAVE
D DIM
F FOR
G GOTO
H GOSUB
J LOAD
K
CAPS SHIFT
Z COPY
X CLEAR
C CONT
V CLS
B BORDER
N NEXT
M PAUSE

MicroTextos

Curso Visual en Pantalla

PROGRAMACION PASO-A-PASO

ZX SPECTRUM



IAN GRAHAM

MicroTextos

LIBRO DOS

SUMARIO

Una edición de
MicroTextos s. a. de ediciones
Paseo de la Castellana 166
28046 Madrid
Teléf. 458 65 91 Telex 49416

Editora Teresa Santiago
Traducción Tratecsa
Editor del proyecto David Burnie
Editor artístico Peter Luff
Diseñador Steve Wilson
Fotografía Vincent Oliver
Director de edición Alan Buckingham
Director artístico Stuart Jackman

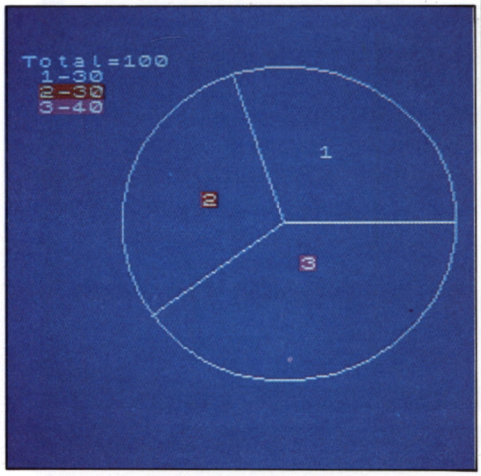
Obra original Step-by-Step Programming ZX Spectrum
Publicada por Dorling Kindersley Ltd., London

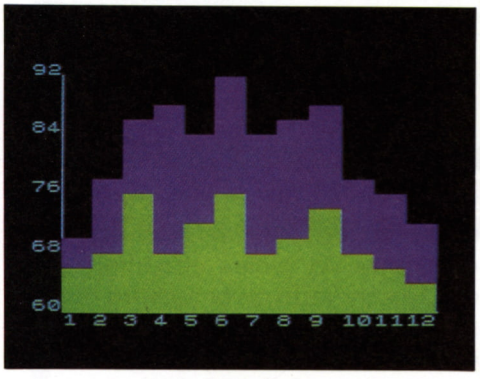
ISBN: 84-86376-07-6
ISBN Obra completa: 84-86376-05-X
Depósito Legal: M-28512-1985

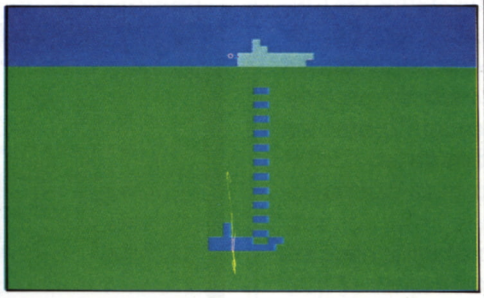
Fotocomposición
Pérez-Díaz, S. A., Madrid

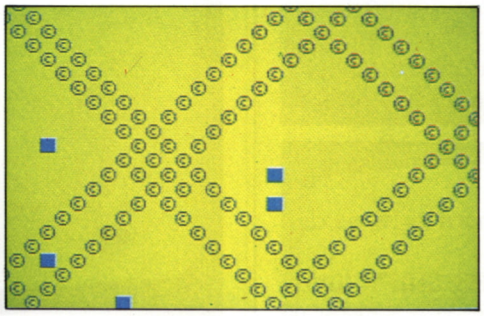
Fotomecánica
Progreso Gráfico, S. A., Madrid

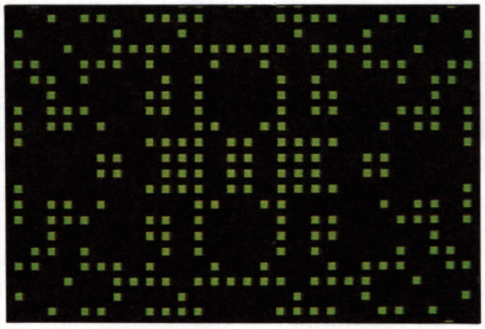
Impresión y Encuadernación
Unigraf, S. A., Madrid

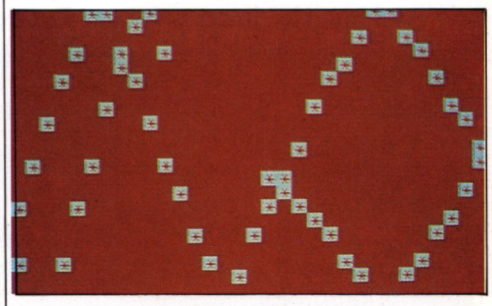












DEFINIR FUNCIONES

Todos los ordenadores disponen de una gama de funciones incorporadas, mandatos que pueden usarse para transformar un número en otro. El resultado de las funciones puede usarse posteriormente en un programa. SQR (Raíz Cuadrada) o INT (Entero) son ejemplos de funciones pre-programadas en el Spectrum. Usando estos mandatos, se toma un número y se opera sobre él para producir otro número.

La gama de funciones del teclado en el Spectrum es muy amplia. Pero si quiere usar una función que no aparece en el teclado, no es necesario cada vez teclear una cadena de instrucciones. Puede programar el Spectrum para realizar secuencias específicas de cálculos. Estas funciones son "llamadas" por el mandato FN (FunciónN) y definidas por el mandato DEF FN (DEFinir FunciónN).

Cómo escribir una función

Para usar una función, primero debe definir qué ha de hacer. Esto se consigue con una sentencia de definición. Por ejemplo:

```
120 DEF FN a(x)=4*x+36
```

define una función llamada "a". El número sobre el que opera una función se llama su argumento. Aquí, el argumento de la función es x. La función primero multiplica por 4 cualquier valor dado de x y luego le suma 36. Si quiere poner en un programa el número 10 de la función, tendría que hacerlo usando la palabra clave FN, así:

```
200 PRINT FN a(10)
```

Esto imprimiría el valor de la función para x=10, es decir 4*10+36, o sea 76.

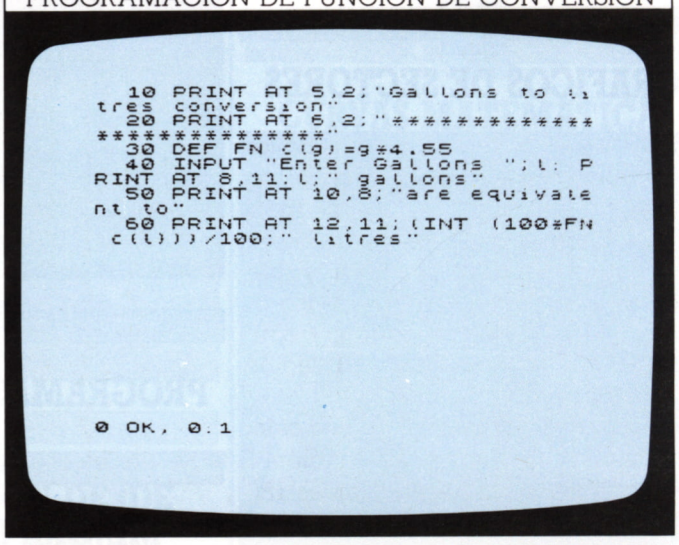
DEF FN se obtiene pulsando CAPS SHIFT junto con SYMBOL SHIFT, seguido de nuevo por SYMBOL SHIFT a la vez que la tecla 1. Con el mismo procedimiento, pero empleando la tecla 2 en vez de 1, se produce FN.

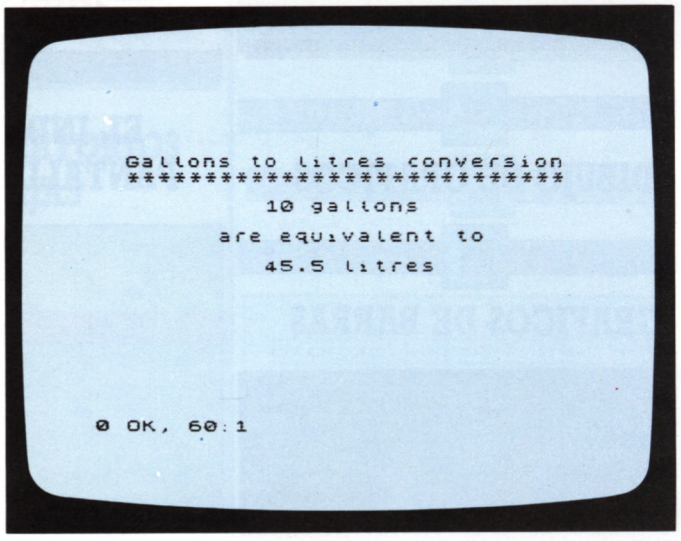
Una vez definida una función en un programa, Vd. puede usar ésta y su argumento como cualquier otro número o variable numérica. Por ejemplo, se pueden sumar, restar, multiplicar y dividir funciones y sus argumentos a la vez, e incluso hacer que las funciones operen sobre números que han sido producidos por funciones. A menos que haga investigación matemática es poco probable ir más lejos, pero para tareas más sencillas las funciones son fáciles de usar y útiles para hacer más sencillos los programas.

¿Por qué usar una función?

El siguiente programa muestra una manera sencilla de poner a trabajar funciones para producir un resultado numérico que después se imprime.

PROGRAMACIÓN DE FUNCIÓN DE CONVERSIÓN





El programa realiza una conversión. La función que realmente hace la conversión está definida en la línea 30. La línea 40 espera a que Vd. teclee el número de galones, que entonces se convierte en el número equivalente de litros por FNc(1) en la línea 60. Multiplicar y dividir por 100 puede parecer extraño, pero es una forma de producir una respuesta con dos decimales con INT. Si usa INT sobre el valor sin multiplicar, se eliminarían los decimales, reduciéndose así la exactitud.

Tomarse la molestia de usar una función aquí, puede parecer innecesario y de hecho es improbable que use FN en un programa sencillo. Pero imagine lo que sucedería si quisiera hacer el cálculo varias veces en diversos lugares de un programa y con números diferentes. Cuando la función es larga y complicada, el definirla una vez le permite escribir y comprobar las líneas de cálculo más sencillamente. FN es como una subrutina de un mandato que sólo trata con números.

Como en realidad una expresión que contiene FN representa un número, Vd. puede usarla para sustituir cualquier clase de cálculo complejo. Cuando escribe sus propias funciones, realmente está dando al ordenador funciones que su lenguaje de programación residente (BASIC) aún no tiene; extendiendo las capacidades del lenguaje.

Secuencias de cálculo con funciones

Imagine que quiere calcular el coste de algo que se vende por superficie —quizá moqueta para una casa. Necesitaría multiplicar el ancho y largo de cada habitación para obtener su área, y luego multiplicarlo por el coste de moqueta por metro cuadrado. Si designa con X e Y el largo y ancho, y con C el coste unitario, entonces el coste por habitación se calcularía por (X*Y)*C.

En el siguiente programa se calcula el coste de cada habitación mediante una función. Ésta se define en la línea 190, y se usa en las líneas 180 y 200:

PROGRAMA CÁLCULO DE MOQUETAS

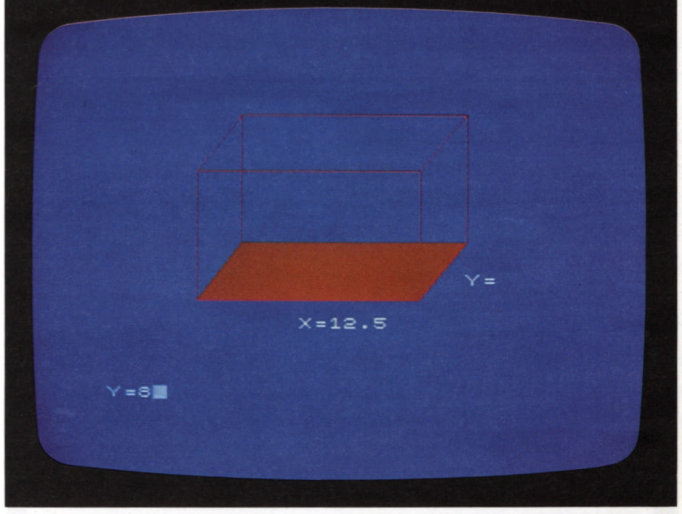
```
  10 LET A=0
  20 BORDER 1: PAPER 1: INK 2: C
LS
  30 PLOT 72,144
  40 DRAW 120,0: DRAW -24,-32: D
RAW -120,0: DRAW 24,32: DRAW 0,-
72: DRAW 120,0: DRAW -24,-32: DR
AW -120,0: DRAW 24,32
  50 PLOT 192,72: DRAW 0,72: PLO
T 168,40: DRAW 0,72: PLOT 48,40:
 DRAW 0,72
  60 FOR X=48 TO 168
  70 PLOT X,40
  80 DRAW 24,32
  90 NEXT X
 100 INK 7
 110 PRINT AT 18,13;"X=";AT 15,2
4;"Y="
 120 INPUT "X=";X: PRINT AT 18,1
5;X
 130 INPUT "Y=";Y: PRINT AT 15,2
6;Y

scroll?
```

```
 140 PAUSE 150: PAPER 0: BORDER
0: CLS
 150 PRINT AT 6,3;"LENGTH= ";X;A
T 8,3;"WIDTH= ";Y
 160 INPUT "COST PER SQUARE METR
E? ";C
 170 PRINT AT 4,3;"COST PER SQUA
RE METRE= ";C
 180 PRINT AT 12,3;"COST= ";FN T
(C)
 190 DEF FN T(C)=(X*Y)*C
 200 LET A=A+FN T(C)
 210 PRINT AT 14,3;"TOTAL COST=
";A
 220 PAUSE 250
 230 GO TO 20



0 OK, 0:1
```

Las líneas 20 a 130 crean una pantalla gráfica que produce un boceto de la habitación, y después espera a que Vd. introduzca (ENTER) valores para el ancho y largo. Una vez hecho esto, el programa introduce (INPUT) los valores y después borra la pantalla.

La siguiente pantalla le pregunta el precio de la moqueta por metro cuadrado. Una vez tecleado, el programa le da el coste de enmoquetar la habitación. La función T da este coste y en la línea 200 se usa para actualizar los costes totales. Si Vd. guarda los valores tecleados cada vez que el programa vuelve a la primera pantalla, encontrará que la línea de Coste Total (TOTAL COST) en la segunda pantalla está actualizada para indicar el coste total de todos los costes calculados:

PANTALLAS CÁLCULO DE MOQUETAS




```
COST PER SQUARE METRE= 6.5
LENGTH= 12.5
WIDTH= 8

COST= 650
TOTAL COST= 1750
```

Vd. puede definir una función en cualquier punto de un programa, pero si las líneas DEF FN y FN están muy separadas en un programa largo, aumentará el tiempo de ejecución. En el programa anterior, usar una función hace que las líneas 180 y 200 sean menos complicadas de lo que serían de otra manera. Ya verá cómo el uso de una función simplifica la confección de gráficos.

AMPLIACION DE DECISIONES

Las palabras de BASIC IF y THEN permiten que un programa opere de una manera, hasta que se encuentra la condición especificada por la sentencia IF. Entonces, el programa es activado para seguir otro curso de acción. Pero la capacidad de IF... THEN no se limita a introducir una sencilla decisión de tipo "sí" o "no". Combinando IF... THEN con las palabras clave AND y OR puede hacer que acometa situaciones mucho más complicadas. Como el BASIC se ha diseñado para reflejar cómo se usan las palabras en el lenguaje corriente del inglés, puede usar IF... THEN justo como lo haría en inglés al describir una serie de condiciones. Aquí tiene un programa que le indica cómo puede llenar la toma de decisión IF... THEN a un nivel más avanzado:

PROGRAMA DE REBOTE

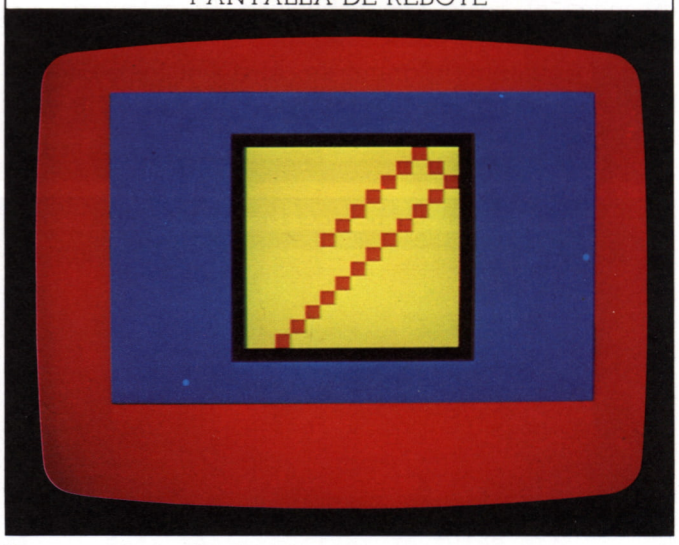
```
 10 BORDER 2: PAPER 1: INK 0: C
LS
 20 FOR r=3 TO 18
 30 PRINT AT r,8;"■"
 40 PRINT AT r,23;"■"
 50 PRINT AT 3,r+5;"■"
 60 PRINT AT 18,r+5;"■"
 70 NEXT r
 80 PAPER 6: INK 2
 90 FOR r=4 TO 17
100 FOR c=9 TO 22
110 PRINT AT r,c;" "
120 NEXT c
130 NEXT r
140 LET r=5+INT (RND*12)
150 LET c=10+INT (RND*12)
160 LET a=1: LET b=1
170 PRINT AT r,c;" "
180 LET r=r+a
190 LET c=c+b
200 IF r=4 OR r=17 THEN LET a=-
a

scroll?
```

```
210 IF c=9 OR c=22 THEN LET b=-
b
220 PRINT AT r,c;"■"
230 BEEP 0.1,3*r
240 GO TO 170









0 OK, 0:1
```

Las líneas 10 a 130 simplemente establecen la pantalla. Las líneas 140 y 150 producen una posición inicial al azar de un «balón» (un cuadrado) dentro del cerco. Para simular el movimiento del balón, la línea 170 borra la imagen en r,c y las líneas 180 y 190 producen nuevas coordenadas en las que la línea 220 revisualiza el balón.

Antes de que suceda esto, las líneas 200 y 210 comprueban si el balón ha alcanzado alguna de las paredes del cerco. Examinan la posición del balón para ver si el número de fila es uno inferior a la tapa del cerco o uno superior al fondo del cerco, o si su número de columna es uno más que el lateral izquierdo o uno menos que el derecho. Si se satisface cualquiera de estas condiciones, las líneas 200 o 210 invierten el movimiento vertical u horizontal del balón:

PANTALLA DE REBOTE



Suma conjunta de decisiones

Ahora puede dar un paso más allá del programa anterior para ver cómo puede incorporar una segunda opción en una sentencia IF... THEN. El siguiente programa describe dos balones que se mueven independientemente:

PROGRAMA DE DOBLE REBOTE (1)

```
 10 BORDER 2: PAPER 1: INK 0: C
LS
 20 FOR r=3 TO 18
 30 PRINT AT r,8;"■";AT r,23;"■
"
 40 PRINT AT 3,r+5;"■";AT 18,r+
5;"■"
 50 NEXT r
 60 PAPER 6: INK 2
 70 FOR r=4 TO 17
 80 FOR c=9 TO 22
 90 PRINT AT r,c;" "
100 NEXT c
110 NEXT r
120 LET r=5+INT (RND*12)
130 LET s=5+INT (RND*12)
140 LET c=10+INT (RND*12)
150 LET d=10+INT (RND*12)
160 LET a=1: LET b=1
170 LET k=-1.3: LET l=1
180 PRINT AT r,c;" "
190 PRINT AT s,d;" "

scroll?
```

Este programa es muy similar al primero excepto que ahora hay dos líneas que visualizan (PRINT) un cuadrado en números cambiantes de fila y columna. Como antes, cada cuadrado empieza en una coordenada al azar que está definida en las líneas 120 a 150. Los cuadrados son animados por las líneas 160 a 300:

PROGRAMA DE DOBLE REBOTE (2)

```
 200 LET r=r+a: LET c=c+b
 210 LET s=s+k: LET d=d+l
 220 IF r=4 OR r=17 THEN LET a=-
a
 230 IF s<5 OR s>16 THEN LET k=-
k
 240 IF c=9 OR c=22 THEN LET b=-
b
 250 IF d=9 OR d=22 THEN LET l=-
l
 260 PRINT AT r,c;"■"
 270 PRINT AT s,d;"■"
 280 IF r>=INT (s+0.5)-1 AND r<=
INT (s+0.5)+1 AND c>=d-1 AND c<=
d+1 THEN BEEP 0.5,-10: GO TO 10
 290 BEEP 0.02,3*r: BEEP 0.02,3*
s
 300 GO TO 180


0 OK, 0:1
```

Se hace que el segundo balón salga en dirección diferente del primero y a una velocidad vertical ligeramente superior, de forma que los dos balones tengan mayor probabilidad de encontrarse. De otra forma se seguirían el uno al otro alrededor del cerco en las mismas pistas. En la línea 280 el ordenador toma una decisión múltiple acerca de la posición de ambos cuadrados. Sin esta línea, cuando se encontraran los cuadrados, simplemente seguirían su camino como si nada hubiese sucedido. Esta no es una simulación muy convincente de lo que realmente ocurriría, por lo que la línea 280 decide si los cuadrados están lo suficientemente próximos como para chocar. La línea incluye una decisión IF... THEN con tres AND para ver si r y s están bastante juntos y si c y d lo están también dentro de los mismos límites. Esto lo hace tomando r, por ejemplo, y decidiendo si es mayor o igual que INT(s+0.5)−1 y simultáneamente menor o igual que INT(s+0.5)+1.

Si se cumplen todas estas condiciones, significa que los dos balones o bien ocupan el mismo cuadrado o son adyacentes, en cuyo caso se supone que han colisionado. Suena un BEEP y todo el proceso empieza de nuevo.

IF... THEN en programación de juegos

La toma de decisión por este método es muy útil si Vd. quiere saber si dos caracteres ocupan o no la misma posición de pantalla. Se suele usar en programas donde un carácter es "barrido" por otro. El siguiente programa muestra cómo puede incorporar la técnica en un juego sencillo que tiene un número de caracteres moviéndose a la vez:

PROGRAMA DE BASE DE DISPARO

```
 10 BORDER 1: PAPER 0: INK 6: C
LS
 20 PRINT AT 19,15;"▚ ▞";AT 20,
16;"▀";AT 21,14;"▟▀▙"
 30 LET c=0: LET d=7: LET r=18
 40 PRINT AT 3,c;" ";AT 7,d;"
 50 LET c=c+1: LET d=d+1
 60 IF c>31 THEN LET c=0
 70 IF d>31 THEN LET d=0
 80 PRINT INK 7;AT 3,c;">";AT 7
,d;">"
 90 PRINT AT r,16;" "
100 IF r=0 THEN LET r=19
110 LET r=r-1
120 PRINT AT r,16;"↑": BEEP 0.0
2,800/(3*r+15)
130 IF r=3 AND c=16 OR r=7 AND
d=16 THEN BEEP 0.5,r↑2: GO TO 10
140 GO TO 40


0 OK, 0:1
```

El programa visualiza una base de disparo en el fondo de la pantalla. Dispara hacia arriba flechas contra dos flechas horizontales que cruzan la pantalla. La línea 130 comprueba si las coordenadas de la flecha ascendente son iguales a alguna de las dos horizontales. Si lo son, el programa retrocede a la línea 10 y empieza de nuevo. Si no, retrocede a la línea 40 y mueve todos los caracteres un espacio. La línea 100 comprueba si la flecha ascendente ha alcanzado el tope de la pantalla. Si es así, se lanza una nueva flecha desde la base de disparo. Las líneas 60 y 70 comprueban si alguna de las flechas horizontales ha alcanzado el borde derecho de la pantalla. Si es así, su coordenada de columna se restaura a cero.

Cuando procesa el programa, deberá encontrar que la flecha de la base de disparo encuentra su blanco en el séptimo paso de la flecha a través de la pantalla. Esto sucede porque el programa está trabajando con números fijados. Si en vez de esto utiliza RND (aleatorio), los resultados se vuelven impredecibles.

VISUALIZACIÓN DE BASE DE DISPARO

CAMBIO DE PASO EN GRAFICOS

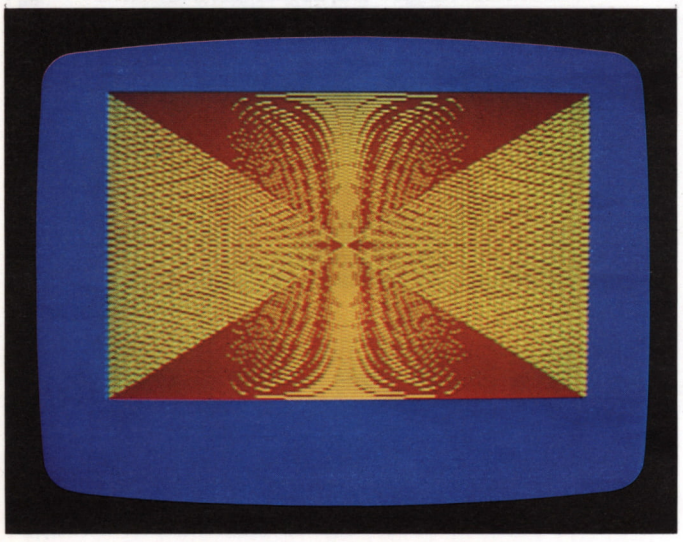
Los bucles FOR... NEXT son muy útiles para repetir una secuencia de sentencias de programa un número predecible de veces. Si quiere aumentar el valor de una variable en más de 1 en cada ciclo en un bucle, deberá reemplazar FOR... NEXT por una sentencia GOTO precedida por N=N+2 o cualquier cambio que se quiera dar a N. Sin embargo, si ya ha empezado a escribir un programa y encuentra que el bucle FOR... NEXT es inadecuado, a veces es difícil y requiere tiempo sustituir una técnica de programación completamente nueva como ésta. Pero hay una forma mucho más sencilla de saltar adelante o atrás en un bucle.

Cómo cambiar los tamaños de salto con STEP

La palabra clave del BASIC para esto es STEP, que puede ya haber encontrado en el Libro 1. Aquí hay un programa que usa STEP con gráficos:

GRÁFICOS CON **STEP**

```
LS
  10 BORDER 1: PAPER 6: INK 2: C
  20 FOR y=0 TO 175 STEP 4
  30 PLOT 0,0
  40 BEEP 0.1,y/3
  50 DRAW 255,y
  60 PLOT 255,0
  70 DRAW -255,y
  80 PLOT 0,175
  90 DRAW 255,-y
 100 PLOT 255,175
 110 DRAW -255,-y
 120 NEXT y

0 OK, 0:1
```



El cursor de gráficos se mueve a cada una de las cuatro esquinas de la pantalla, una detrás de otra. Se dibuja una línea desde cada esquina a un punto del lado opuesto de la pantalla. La separación de líneas está determinada por el tamaño STEP en la línea 20. STEP 4 da los mejores resultados. Esto hace que los valores de y sean 0, 4, 8, 12, 16, etc., en vez de que y aumente en 1 en cada circuito del bucle.

Uso de STEP para dibujar (DRAW) redes

Una pantalla que puede obtener usando STEP es una red para juegos. Puede obtenerla dibujando una serie de líneas entrecruzadas sobre un fondo de contraste. Pero en vez de especificar cada línea, puede usar bucles con STEP que le hagan la mayoría del trabajo:

PROGRAMA DE RED DE JUEGOS

```
LS
  10 BORDER 6: PAPER 2: INK 7: C
  20 FOR y=25 TO 151 STEP 21
  30 PLOT 26,y
  40 DRAW 204,0
  50 NEXT y
  60 FOR x=26 TO 230 STEP 34
  70 PLOT x,25
  80 DRAW 0,126
  90 NEXT x

0 OK, 0:1
```

Las líneas 20 a 50 dibujan una serie de líneas a través de la pantalla a intervalos verticales de 21. Después las líneas 60 a 90 dibujan líneas verticales con separación 34. (Las coordenadas y separaciones pueden verse mejor si hace referencia al gráfico de la red, de la página 59.)

El siguiente programa produce una red de juegos con tantas cuadrículas como quiera. El programa le pregunta qué clase de red quiere (cuántas cuadrículas en horizontal y vertical) e introduce esos números en la subrutina que permite un modelo generalizado.

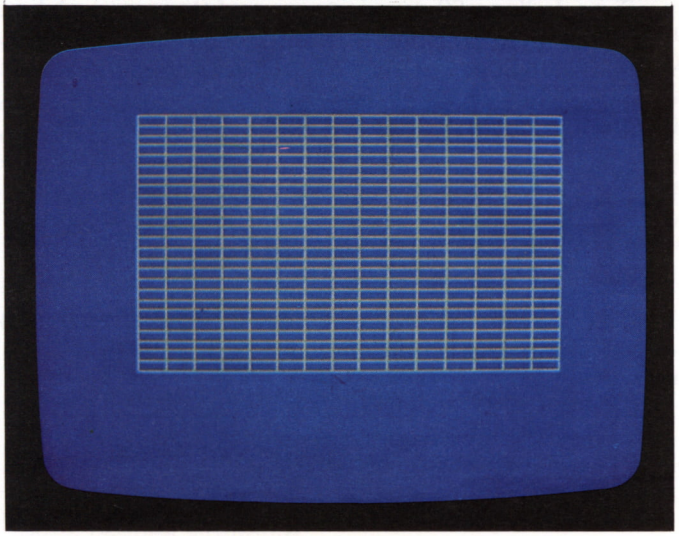
El programa se divide en tres secciones. La primera, de entrada (INPUT) (líneas 10 a 100), le pide la información necesaria para la segunda parte del programa. Esta, con los números introducidos, calcula las separaciones entre las líneas dibujadas en la tercera sección (líneas 140 a 240). Esto se consigue con la variable w, que representa el número de cuadrículas a lo ancho, que se divide con el ancho de la pantalla en la línea 130; y la h, que representa el número de cuadrículas

de arriba abajo dividido por la altura de la pantalla. Con este programa puede obtener una gama casi ilimitada de redes de cuadrados o rectángulos, según los valores introducidos:

PROGRAMA DE REJILLA VARIABLE

```
 10 BORDER 1: PAPER 7: INK 0: C
LS
 20 PRINT AT 9,9;"Do It Yoursel
f"
 30 PRINT AT 11,11;"Games Grid"
 40 PAUSE 150
 50 CLS
 60 PRINT AT 9,6;"Enter the gri
d size"
 70 PRINT AT 11,4;"How many squ
ares across?"
 80 INPUT w
 90 PRINT AT 13,5;"How many squ
ares high?"
100 INPUT h
110 PAPER 1: INK 6: CLS
120 LET x=16: LET y=16
130 LET a=224/w: LET b=144/h
140 FOR n=1 TO w+1
150 PLOT x,y
160 DRAW 0,144
170 LET x=x+a

scroll?
```

```
180 NEXT n
190 LET x=16: LET y=16
200 FOR n=1 TO h+1
210 PLOT x,y
220 DRAW 224,0
230 LET y=y+b
240 NEXT n



0 OK, 0:1
```



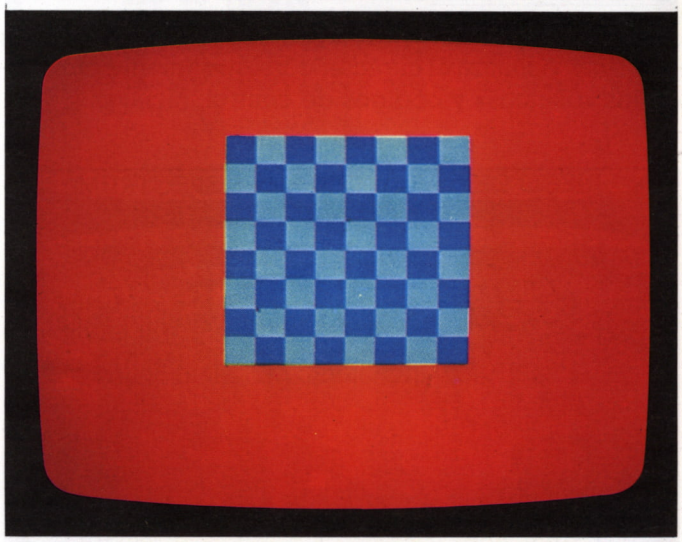
Puede usar este programa como base para obtener un gráfico y después visualizar valores en el interior de cada cuadrado o rectángulo. Usando la técnica de matrices (págs. 22-23), puede hacer que los valores del gráfico reaccionen con diferentes entradas.

Cómo visualizar un tablero de ajedrez

Para conseguir que el ordenador visualice un ajedrez, primero puede visualizar cuadrados alternativos blancos y negros, trabajando de arriba abajo, y después simplemente añadir un color a la pantalla con los mandatos BORDER, PAPER, e INK:

PROGRAMA DE TABLERO DE AJEDREZ

```
 10 PAPER 2: BORDER 2: INK 5: C
LS
 20 FOR r=3 TO 18
 30 FOR c=8 TO 23
 40 PRINT AT r,c;"■"
 50 NEXT c
 60 NEXT r
 70 INK 1
 80 FOR r=3 TO 15 STEP 4
 90 FOR c=8 TO 20 STEP 4
100 PRINT AT r,c;"■■"
110 PRINT AT r+1,c;"■■"
120 PRINT AT r+2,c+2;"■■"
130 PRINT AT r+3,c+2;"■■"
140 NEXT c
150 NEXT r


K
```



Aquí PRINT AT se usa para producir hileras de cuadrados azul oscuro sobre un fondo cyan. El programa selecciona un color azul (INK 1) y fija la coordenada de la fila más alta del tablero. Habiendo visualizado un cuadrado oscuro, se salta el siguiente, dejándolo en cyan, y visualizando otro oscuro en la tercera posición. El tamaño STEP de columna se fijó en 4 para esto. Como cada bucle dibuja dos hileras de cuadrados, el tamaño 4 de STEP de fila salta cada fila.

INFORMACION DESDE EL TECLADO

Para teclear una información mientras se procesa un programa Vd. ha usado —hasta ahora— el mandato INPUT. Con éste, debe pulsar ENTER después de teclear la información. Esto tiene sus inconvenientes. Incluso cuando Vd. sabe que debe pulsar ENTER, puede olvidarlo y usando dos teclas en secuencia también retrasa el programa. Es mucho más útil que el ordenador responda sin esperar el pulsado de ENTER, como las máquinas de los bares responden al pulsado de un botón. Para hacer que el Spectrum haga esto use la palabra clave INKEY$.

Códigos ASCII

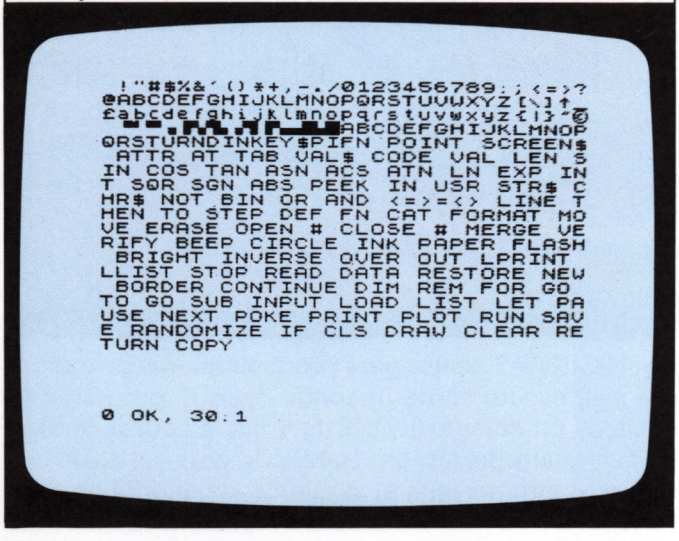
Todos los símbolos reconocidos por el ordenador —palabras clave, números, letras, signos de puntuación y matemáticos— se almacenan en su interior como números, del 0 al 255. La mayoría de los ordenadores almacenan estos números según un código llamado ASCII (American Standard Code for Information Interchange). ASCII almacena la letra A como 65, el signo + como 43 y el número 1 como 49. (Ver en pág. 60 la tabla completa del ASCII.)

Puede ver todos los caracteres que usa el Spectrum tecleando el siguiente programa:

```
10 FOR n=33 TO 255
20 PRINT CHR$ n;
30 NEXT n
```

Así, el ordenador toma cada código ASCII uno después de otro y visualiza el carácter correspondiente omitiendo los «blancos» y códigos de control iniciales:

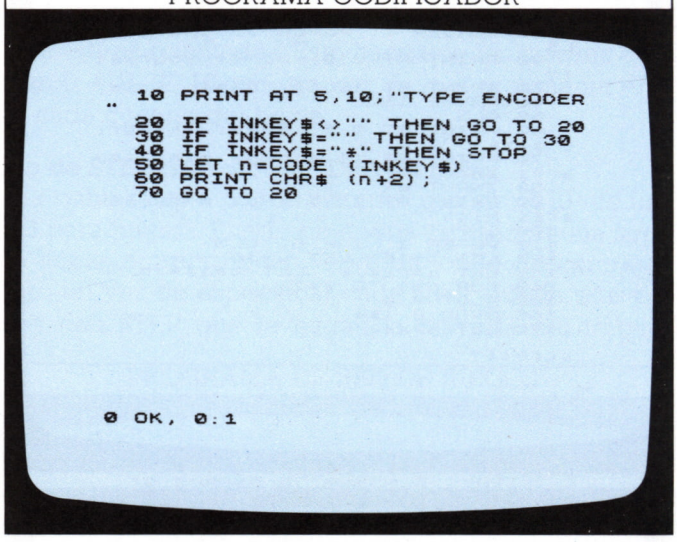
JUEGO DE CARACTERES DEL SPECTRUM



En la figura anterior comprobará que algunos códigos representan sólo una letra o número, mientras otros designan palabras clave completas del BASIC.

El programa siguiente usa INKEY$ para responder a las teclas a medida que se pulsan y después usa números ASCII para visualizar un código simple:

PROGRAMA CODIFICADOR



La línea 10 visualiza el título y las líneas 20 y 30 usan INKEY$ para esperar que Vd. pulse una tecla antes de continuar. La línea 20 puede chocarle. Cámbiela por:

```
20 REM IF INKEY$<>""THEN GOTO 20
```

para que no funcione. Encontrará imposible pulsar una tecla y obtener un solo carácter en la pantalla. Contrariamente a INPUT, INKEY$ no le espera —comprueba si se ha pulsado cualquier tecla y va a la línea siguiente. Así la línea 20 no deja efectuar nada hasta que Vd. haya apretado una tecla, después la línea 30 impide el procesado hasta que Vd. pulse la tecla siguiente.

La línea 40 comprueba después el carácter tecleado para ver si es un asterisco. Si lo es, el programa termina. Si no, la línea 50 convierte su carácter en su código ASCII. La línea 60 suma 2 al código que tecleó y después visualiza el carácter equivalente. Así, el programa parece visualizar disparates cuando teclea algo. No tiene sentido producir un programa codificador si no se puede descifrar el código. Puede transformar el programa codificador en uno decodificador insertando:

```
10 PRINT AT 5,10; "TYPE DECODER"
60 PRINT CHR$(n-2);
```

Para ver si trabaja intente teclear lo siguiente cuando el programa esté en proceso:

```
ncgtp"vq"rtqitco"ykvj"vjg"FM"uetggp"ujqv"ugtkgu
```

Comprobación de sus reacciones con INKEY$

Vd. puede usar la capacidad de INKEY$ para comprobar el teclado, en combinación con los códigos ASCII,

para escribir un programa que le permita cronometrar la velocidad de sus reacciones.

Este programa produce un símbolo al azar y después mide el tiempo que tarda Vd. en pulsarlo.

COMPROBADOR DE REACCIONES

```
  10 BORDER 7: PAPER 7: INK 0: C
LS
  20 PRINT AT 8,7;"Test your ref
lexes"
  30 PRINT AT 10,10;"against the
"
  40 PRINT AT 12,8;"REACTION TES
TER"
  50 PAUSE 150: CLS
  60 PRINT AT 11,9;"Find this ke
y"
  70 LET k=33+RND*94: LET n=0
  80 BEEP 0.1,30
  90 PRINT AT 13,16;CHR$ (k)
 100 LET n=n+1
 110 IF INKEY$<>CHR$ (k) THEN GO
 TO 90
 120 CLS
 130 PRINT AT 11,5;"You took ";(
INT ((n*2.68))/100;" seconds"


0 OK, 0:1
```

```
You took 1.25 seconds



0 OK, 130:1
```

Las líneas 10 a 50 visualizan el título y hacen una pausa de 3 segundos. Después comienza el juego. La línea 70 fija k igual a un número al azar entre el 33 y el 127, los números de código ASCII para las letras y símbolos del Spectrum (excluye los símbolos gráficos y las palabras clave). La línea 70 también fija n igual a cero —n se usa después para calcular el tiempo empleado en pulsar la tecla correcta. La línea 80 hace sonar un "beep" al comienzo del tiempo. La línea 90 visualiza el carácter equivalente al código de azar encontrado por la 70. La línea 110 comprueba si el tecleado ha sido correcto. Si no, vuelve a la 90, incrementando n en 1 en cada bucle.

Si el tecleado fue correcto, se calcula el tiempo desde el "BEEP" por (INT(n*2,68))/100 seg. Si quiere saber por qué ésta es la línea a usar, intente insertar en el programa la línea siguiente:

```
105 IF n=1000 THEN BEEP 0.3,20
```

Esto permite que cronometre 1000 bucles con su reloj. Se tarda aproximadamente 26,85 seg.; cada bucle 0.02685 seg. Para hallar el tiempo empleado, multiplique n por 0.02685. Para limitar el tiempo visualizado por la línea 130 a dos decimales, multiplique (n*0.02685) por 100(=n*2.685), tome su valor entero (para desechar las cifras posteriores a la coma decimal) y finalmente divida el resultado por 100.

Uso de INKEY$ para controlar el movimiento

Las rápidas respuestas de INKEY$ lo hacen ideal para controlar el movimiento sobre la pantalla. En el siguiente programa en vez de ir a través de series predeterminadas de movimientos, el ordenador espera hasta que se pulsen la Z o la M y después mueve el carácter a la izquierda o derecha respectivamente. Las líneas 170 y 180 paran el movimiento del carácter más allá del borde de la pantalla:

PROGRAMA DE ANIMACIÓN **INKEY$**

```
  10 BORDER 1: PAPER 2: INK 6: C
LS
  20 DATA 0,0,59,220,0,0,25,152,
64,2,15,240,32,4,63,252
  30 DATA 18,72,79,242,11,208,95
,250,13,176,129,129,31,248,131,1
93
  40 FOR n=0 TO 7
  50 READ a,b,c,d
  60 POKE USR "e"+n,a
  70 POKE USR "f"+n,b
  80 POKE USR "g"+n,c
  90 POKE USR "h"+n,d
 100 NEXT n
 110 LET c=16
 120 GO TO 210
 130 IF INKEY$<>"" THEN GO TO 13
0
 140 IF INKEY$="" THEN GO TO 140
 150 IF INKEY$="z" THEN LET c=c-
1
 160 IF INKEY$="m" THEN LET c=c+

scroll?
```

```
 170 IF c<0 THEN LET c=0
 180 IF c>30 THEN LET c=30
 190 PRINT AT 10,c-1;"  "
 200 PRINT AT 11,c-1;"  "
 210 PRINT AT 10,c;"EF"
 220 PRINT AT 11,c;"GH"
 230 PAUSE 5
 240 GO TO 130









0 OK, 0:1
```

Si Vd. procesa el programa verá que el carácter responde sólo a las teclas Z y M.

NUMEROS ALEATORIOS Y VENTAJAS

El Spectrum usa dos palabras clave para tratar los números aleatorios: RND y RANDOMIZE. Como se vio, RND se usa para generar un número aleatorio. Una línea como la siguiente:

```
80 LET a=RND
```

generará un número aleatorio entre 0 y 0.99999999.

En realidad, esto no es exactamente así. El Spectrum tiene almacenados en memoria 65.536 números diferentes entre 0 y 0.99999999 mezclados, sin modelo definido. Por ello se dice que RND es una función "pseudo-aleatoria". Aunque hay una secuencia para los números, para la mayoría de los fines pueden considerarse completamente aleatorias. Como el margen considerado no es de mucho uso se usa una línea como

```
100 n=1+INT(2*RND)
```

para obtener números aleatorios enteros en otro margen.

La línea anterior produce un 1 o un 2, quizás para un programa de "cara o cruz". 2*RND produce un número entre 0 y 1.99999998, e INT lo redondea al menor entero (0 ó 1) y sumando 1 se obtiene 1 ó 2. Análogamente,

```
100 n=1+INT(6*RND)
```

produce números entre 1 y 6.

Cómo iniciar una secuencia "aleatoria"

La depuración (eliminación de errores) de un programa que usa RND puede ser difícil ya que dos RNDs no producen el mismo resultado; es más fácil hallar los errores si las RNDs producen el mismo número cada vez que se procesa el programa. Teclee primero

PROGRAMA **RND**

```
 120 PRINT AT 5,3;"PREDICTING RA
NDOM NUMBERS"
 130 FOR n=1 TO 6
 140 PRINT AT 2*n+6,11;14*RND
 150 NEXT n
```

Este sencillo listado produce una serie de seis números aleatorios. En cada proceso del programa debería tener una serie diferente de números. Sin embargo la pantalla siguiente produce una serie siempre idéntica, obtenida añadiendo la línea:

```
125 RANDOMIZE 10
```

El uso de RANDOMIZE (aparece como RAND en el teclado) produce en cada proceso los mismos números; trabaja controlando dónde inicia RND la selección entre sus 65.536 números almacenados.

PANTALLAS REPETICIÓN/AZAR

```
PREDICTING RANDOM NUMBERS

          4.5458984
          4.953064
          7.4900513
          1.7611084
          6.0970154
          9.2851257


0 OK, 150:1
```

```
PREDICTING RANDOM NUMBERS

          0.17602539
          0.17602539
          0.17602539
          0.17602539
          0.17602539
          0.17602539


0 OK, 150:1
```

La segunda pantalla muestra lo que sucede al mover RANDOMIZE. Usando el editor, cambie la línea 125 al número de línea 135 y borre 125. Debería obtener el mismo número (0.17602539) cada vez. RANDOMIZE 10 inicia la secuencia aleatoria en el décimo número almacenado en cada proceso. Vd. puede seguir RANDO-

MIZE con cualquier número entre 1 y 65.536. RANDOMIZE por su cuenta usa el tiempo desde que se conectó el ordenador para fijar el punto de iniciación para RND. Así, usando RANDOMIZE con RND se hace RND aun más aleatorio.

Cómo establecer ventaja en juegos de aventuras

Haga lo que haga con RND, sólo produce un número cada vez. Esto es bueno para programas de cara o cruz o de dados en que cada resultado posible tiene la misma probabilidad. Pero imaginemos que queremos crear probabilidad en un programa de forma que algunos resultados sean más probables que otros. Así se escriben muchos juegos de aventuras, haciendo los programas más interesantes que los que manejan secuencias predecibles de sucesos.

El siguiente programa demuestra cómo, en un juego de aventura, puede hacerse que el jugador encuentre algunos caracteres o símbolos con más frecuencia que otros. Se puede también fijar la ventaja de forma que lo más frecuente es que no suceda nada:

PROGRAMA DE VENTAJA EN JUEGOS

```
  10 DATA 0,0,1,128,1,64,123,128
,0,0,88,192
  20 DATA 0,0,3,192,3,160,123,12
8,0,0,153,224
  30 DATA 0,0,7,224,1,144,122,12
8,0,64,31,240
  40 DATA 0,0,15,240,1,8,50,128,
12,100,31,240
  50 DATA 7,224,15,240,7,196,2,1
28,18,216,15,240
  60 DATA 3,192,15,240,15,234,2,
128,34,240,15,224
  70 DATA 1,128,7,224,19,144,2,1
28,70,104,7,208
  80 DATA 1,128,1,128,123,128,5,
0,76,68,3,12
  90 FOR n=0 TO 7
 100 READ o,p,q,r,s,t,u,v,w,x,y,
z
 110 POKE USR "a"+n,o
 120 POKE USR "b"+n,p
 130 POKE USR "c"+n,q

scroll?
```

```
 140 POKE USR "d"+n,r
 150 POKE USR "e"+n,s
 160 POKE USR "f"+n,t
 170 POKE USR "g"+n,u
 180 POKE USR "h"+n,v
 190 POKE USR "i"+n,w
 200 POKE USR "j"+n,x
 210 POKE USR "k"+n,y
 220 POKE USR "l"+n,z
 230 NEXT n
 240 BORDER 2: PAPER 6: CLS
 250 PRINT INK 3;AT 5,5;"*******
********"
 260 PRINT INK 3;AT 14,5;"******
**********"
 270 PRINT INK 3;AT 19,14;"*****
"
 280 PRINT INK 0;AT 7,5;"AB";AT
8,5;"CD";AT 7,16;"EF";AT 8,16;"G
H";AT 7,26;"IJ";AT 8,26;"KL"
 290 LET k=0: LET d=0: LET t=0
 300 FOR n=1 TO 100

scroll?
```

```
 300 FOR n=1 TO 100
 310 LET a=INT (20*RND+1)
 320 IF a=1 THEN LET A$="AB": LE
T B$="CD": LET t=t+1
 330 IF a=2 OR a=3 THEN LET A$="
EF": LET B$="GH": LET k=k+1
 340 IF a=4 OR a=5 THEN LET A$="
IJ": LET B$="KL": LET d=d+1
 350 IF a>5 THEN LET A$="**": LE
T B$="**"
 360 PRINT AT 10,16;A$;AT 11,16;
B$
 370 PRINT AT 12,5;t;AT 12,16;k;
AT 12,26;d;AT 17,15;n
 380 BEEP 0.2,15
 390 NEXT n



0 OK, 0:1
```

Las líneas 10 a 230 reprogramen las teclas A a L con los 12 caracteres que componen los tres símbolos del juego —un saco de tesoros, un caballero armado y un dragón. Cada símbolo se compone de cuatro caracteres, los dos primeros encima de los otros.

La línea 310 produce un número entre 1 y 20. Si éste es 1, se elige el símbolo del tesoro. Si es 2 o 3 se elige el caballero, y si es 4 o 5, el dragón. Si el número es otro cualquiera (representado por cuatro asteriscos), no se selecciona nada. Así caballero y dragón tienen doble probabilidad que el tesoro.

Cuando se procesa el programa, se hacen 100 selecciones aleatorias. Los procesos totales de tesoro, caballero y dragón se muestran incorporados en la pantalla. Se puede cambiar las probabilidades relativas de aparición de los tres símbolos cambiando los números en las líneas 320 a 350. Insertando una sentencia PAUSE en el bucle, o aumentando la duración del "BEEP" en la línea 380, se retarda el programa. Al final de cada proceso (RUN), se puede ver con qué frecuencia ha elegido el programa cada uno de los tres caracteres.

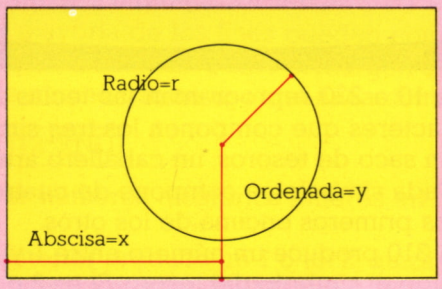
PANTALLA DE VENTAJA EN EL JUEGO

CURVAS Y CIRCULOS

En muchos ordenadores personales el dibujo de curvas y círculos es un proceso muy complicado. En el Spectrum es muy fácil dibujar círculos con el mandato CIRCLE que efectúa los cálculos que se tendrían que hacer en otro caso. Para ello hay que especificar la abscisa x, y la ordenada y, (referidas a la esquina inferior izquierda de la red) y el radio r. La posición y tamaño no tienen más limitaciones que los límites de la rejilla de gráficos.

POSICIONES DE UN CÍRCULO

Con CIRCLE puede producirse cualquier círculo dado por x,y,r



La línea de sentencia se escribe así:

CIRCLE 128,88,50

El centro de este círculo está en el centro de la pantalla (128,88) y el radio es 50 unidades. Así, el diámetro horizontal va desde 78 a 178 unidades y el vertical, de 38 a 138.

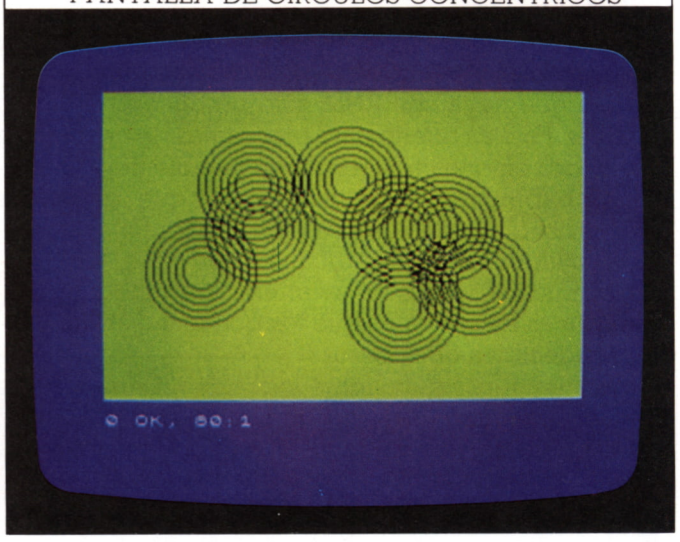
Obtención de modelos con CIRCLE

Se pueden hacer modelos con el mandato CIRCLE seleccionando al azar las coordenadas del centro de un círculo. El programa siguiente construye una pantalla por este método:

PROGRAMAS DE CÍRCULOS CONCÉNTRICOS

```
  10 BORDER 3: PAPER 6: INK 0: C
LS
  20 FOR n=1 TO 8
  30 LET x=50+INT (155*RND)
  40 LET y=50+INT (75*RND)
  50 FOR r=10 TO 30 STEP 4
  60 CIRCLE x,y,r
  70 NEXT r
  80 NEXT n


0 OK, 0:1
```

PANTALLA DE CÍRCULOS CONCÉNTRICOS



Las líneas 30 y 40 producen un par de coordenadas al azar, x e y, tales que ninguna diste menos de 50 de cualquier borde de la pantalla, ya que los círculos han de tener radio 50. El bucle de la línea 50 a 70 dibuja repetidamente círculos concéntricos pero de radio gradualmente creciente (r). La línea 80 inicia de nuevo el proceso entero pero con otro par de coordenadas aleatorias.

Vd. puede cambiar el radio máximo del círculo y el tamaño STEP entre radios cambiando la línea 50:

50 FOR r=10 to 20 STEP 3

o incluso STEP 2, que produce modelos más pequeños.

Dibujo de arcos y ondas

Un arco (parte de una circunferencia) puede obtenerlo en el Spectrum, pero no debe usar CIRCLE para ello. Con el mandato DRAW x,y,a se dibuja un arco de círculo que empieza en el último punto marcado (PLOT) o dibujado, y que termina en el punto de coordenadas x,y. El valor "a" es el más complicado. Es el ángulo barrido por el arco (imaginado como parte de un círculo). Puede ver trabajar el mandato tecleando las dos líneas siguientes (encontrará la tecla PI encima de la M):

PLOT 10,88
DRAW 230,0,PI/4

Esto produce una figura similar a un trozo de cable suspendido por ambos extremos y el centro hundido. La primera línea marca un punto cerca del borde izquierdo en la mitad de altura de la pantalla. La segunda, dibuja un arco desde aquel punto, en sentido contrario a las agujas del reloj, al (10+230),(88+0) o sea 240,88.

Que el arco forme un hundimiento ligero o un semicírculo viene determinado por el valor de "a". Este ángulo se mide en radianes, no en grados. Los radianes son otra forma de medir ángulos, basada en la geometría del círculo. En grados sexagesimales un círculo completo = 360 grados = 2*PI radianes. PI es una constante matemática cuyo valor aproximado es 3,14159625... (véalo tecleando PRINT PI). En la proporción entre la longitud de la circunferencia de un círculo y su diámetro. Naturalmente, se tendrá PI/2 rad.=90 gr.=1/4 de círc. y PI/4=45 gr.=1/8 círc., etc.

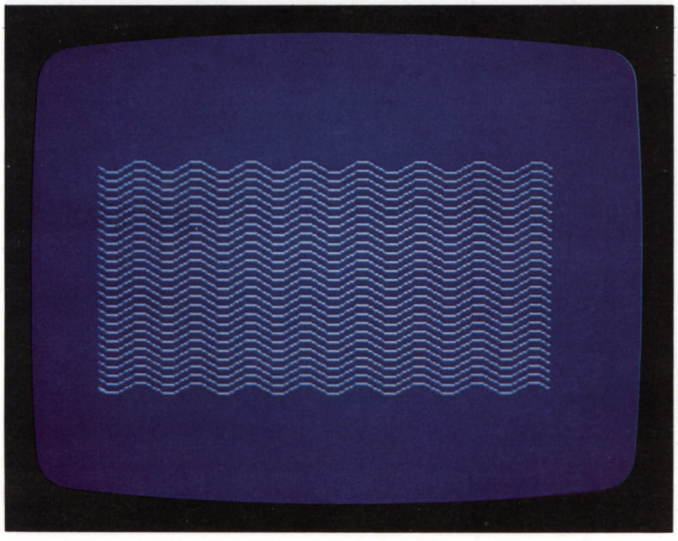
Ahora, sin borrar el primer arco, intente añadir:

```
PLOT 10,88
DRAW 230,0-PI/4
```

El segundo arco se dibuja entre los dos mismos puntos formando una figura como un ojo. Puede producir figuras usando arcos que den modelos de ondas:

PROGRAMA DE MODELO DE ONDAS

```
 10 BORDER 1: PAPER 1: INK 7: C
LS
 20 LET y=128
 30 FOR f=1 TO 26
 40 FOR x=0 TO 225 STEP 30
 50 PLOT x,y
 60 DRAW 15,0,PI/2: DRAW 15,0,-
PI/2
 70 NEXT x
 80 LET y=y-5
 90 NEXT f

0 OK, 0:1
```



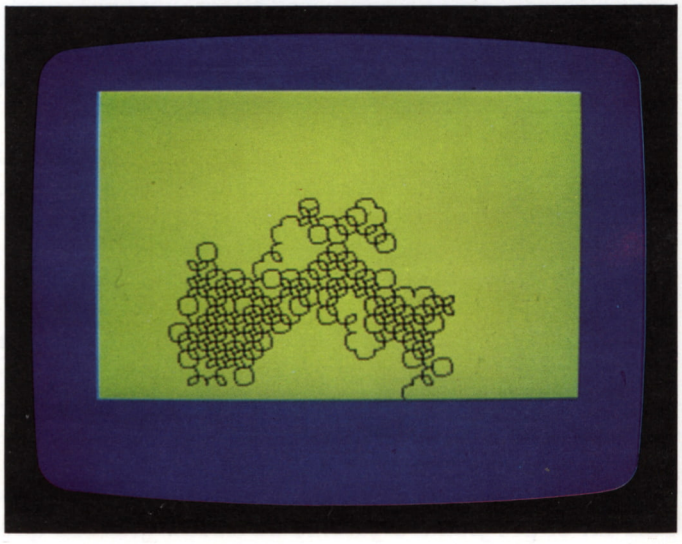
La línea 20 fija la primera ordenada y de las ondas. La línea 40 escalona la abscisa x a través de la pantalla. Es importante que el tamaño STEP sea el mismo que las longitudes x combinadas de las dos partes de la onda formada por cada bucle. La línea 50 marca un punto en x, y fijados al comienzo de cada bucle. La línea 60 dibuja un pequeño arco de círculo y después un segundo arco del mismo tamaño. El primero se curva hacia abajo y el segundo hacia arriba, construyendo una fila de ondas. Cuando ésta se completa, la línea 80 mueve la ordenada y hacia abajo. El programa continúa hasta que las ondas alcanzan la parte baja de la pantalla.

Programación de curvas progresivas

En la página anterior se usó CIRCLE para obtener círculos concéntricos al azar. El programa siguiente dibuja semicírculos al azar, progresivos hacia arriba, abajo, derecha o izquierda, sobre la pantalla.

PROGRAMA DE CURVAS PROGRESIVAS

```
 10 BORDER 1: PAPER 6: INK 0: C
LS
 20 PLOT 128,88
 30 LET x=0: LET y=0
 40 LET q=1+INT (4*RND)
 50 IF q=1 THEN LET x=10
 60 IF q=2 THEN LET x=-10
 70 IF q=3 THEN LET y=10
 80 IF q=4 THEN LET y=-10
 90 DRAW x,y,PI
100 GO TO 30

0 OK, 0:1
```



Intente cambiar el tamaño de arco en las líneas 50 a 80 para efectos diferentes. Verá que si reduce el tamaño a 4, semeja una letra v.

CURVAS MATEMATICAS

La mayoría de las figuras que ha dibujado hasta aquí han implicado resolver antes dónde se encontraban los puntos clave de la figura (esquinas de un cuadrado, por ejemplo), y después dibujar líneas entre ellos. Afortunadamente, pueden dibujarse modelos geométricos más complicados usando funciones que hagan todo el trabajo por Vd; no tiene que marcar ningún punto. Con programas muy cortos puede obtener algunos gráficos llamativos usando las funciones SIN (abreviatura del término matemático seno) y COS (abreviatura de coseno).

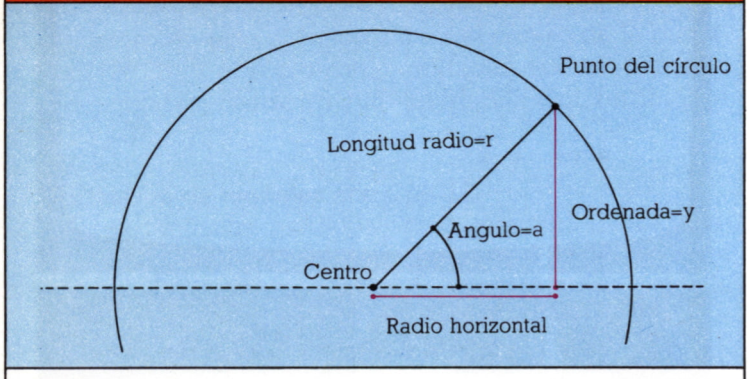
Los programas de estas dos páginas combinan SIN y COS, por lo que conviene saber qué significan en realidad.

Otra manera de dibujar círculos

Puede dibujar círculos sin usar CIRCLE (pág. 16), usando SIN y COS. Aunque el programa es más largo, le permite obtener algunos gráficos más complejos.

Refiriéndonos a una parte de círculo, cada punto de él puede relacionarse con el ángulo que forma.

COORDENADAS DEL CIRCULO



Las distancias x e y pueden expresarse de otra forma: cada ángulo a tiene un valor para su seno y su coseno, y las coordenadas del punto del círculo valen respectivamente:

r*COS(a),r*SIN(a)

Sabido esto, puede hacer que el ordenador dibuje un círculo, marcando todos los valores posibles del ángulo "a", mediante un bucle, el programa utiliza las funciones SIN y COS para dibujar las coordenadas de todos los puntos del círculo.

Se puede hacer esto, más fácilmente utilizando CIRCLE. En el programa siguiente se hacen gráficos que CIRCLE no puede realizar. La línea 20 fija en 80 el radio del círculo. El ángulo ha de variar de cero a 360 grados.

PROGRAMA DE CÍRCULO

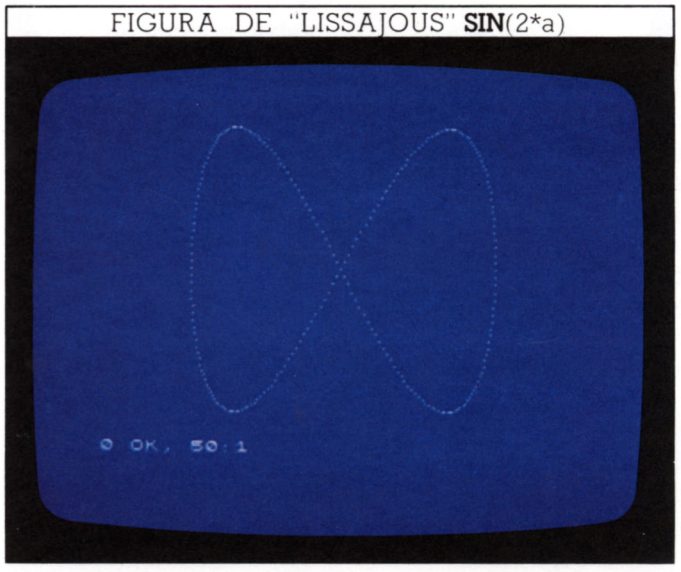
```
  10 BORDER 1: PAPER 1: INK 7: C
LS
  20 LET r=80
  30 FOR a=0 TO 2*PI STEP PI/120
  40 PLOT r*COS (a)+128,r*SIN (a
)+88
  50 NEXT a


0 OK, 0:1
```

Como las funciones SIN y COS del BASIC operan sobre ángulos en radianes, igual que el mandato DRAW x,y,a de la página 16 la gama de ángulos utiliza PI. En la línea 30 se ha elegido STEP PI/120 (línea 30) que da puntos suficientemente juntos, pero acorta el procesado del programa.

Modelos con SIN y COS

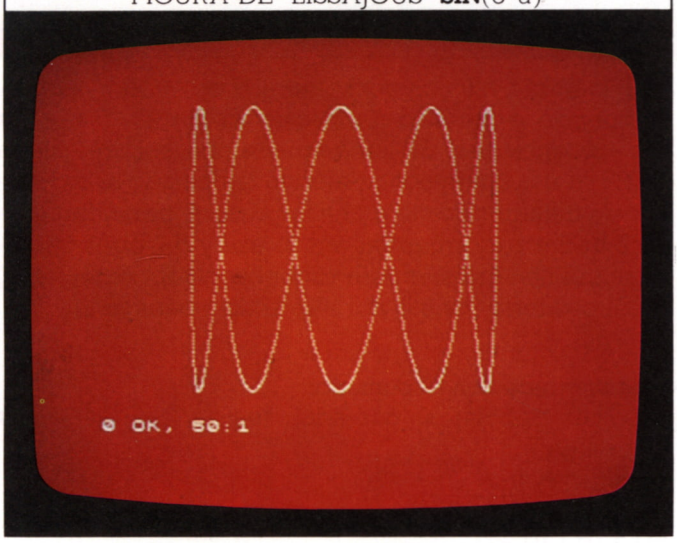
En el programa anterior cuando x=0, y tiene el valor máximo y viceversa. ¿Qué sucede cuando el ángulo afectado por SIN y COS no es exactamente a sino uno deducido de a? Pruebe un ángulo 2*a y después otro 5*a. Obtendrá respectivamente en cada pantalla:

FIGURA DE "LISSAJOUS" **SIN**(2*a)



El número de bucles en cada pantalla es igual al del factor que multiplica "a":

FIGURA DE "LISSAJOUS" **SIN**(5*a)

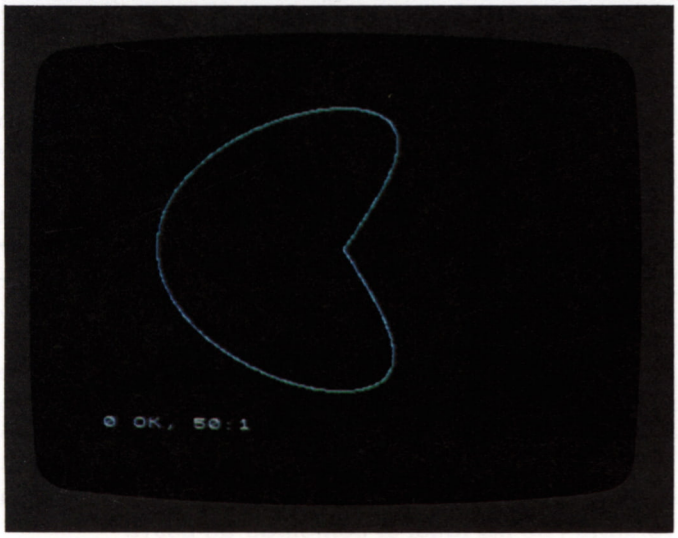


Curvas complejas

Ahora puede hacer otro tipo de cambio:

PROGRAMA DEL RIÑÓN

```
  10 BORDER 0: PAPER 0: INK 5: C
LS
  20 LET r=10
  30 FOR a=0 TO 2*PI STEP PI/480
  40 PLOT (r*COS (a)*r*SIN (0.5*
a))+128,8*r*SIN (a)+88
  50 NEXT a

0 OK, 0:1
```



En este programa se han cambiado los colores y tamaños (STEP) y los valores de las coordenadas. El programa seguirá dibujando si Vd. aumenta el margen de los valores del ángulo. El siguiente programa duplica aproximadamente el margen; también acelera el proceso dibujando líneas cortas en lugar de marcar puntos individuales:

PROGRAMA DEL RELOJ DE ARENA

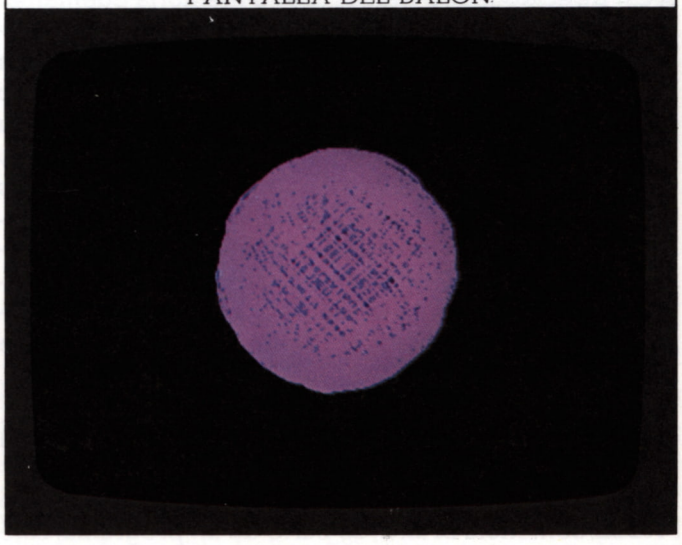
```
  10 BORDER 3: PAPER 2: INK 7: C
LS
  20 LET r=60
  30 LET x=0: LET y=0: PLOT 128,
88
  40 FOR a=0 TO 12.6 STEP 0.2
  50 LET i=r*COS (a)*SIN (0.5*a)
  60 LET j=r*SIN (a)
  70 DRAW i-x,j-y
  80 LET x=i: LET y=j
  90 NEXT a

0 OK, 0:1
```

Pruebe con este programa y obtendrá una gran variedad de formas complicadas. El dibujo siguiente se produce cambiando los colores y líneas 40 y 50 así:

```
40 FOR a=0 TO 1000 STEP 0.1
50 LET i=r*COS(a)*SIEN(0.98 *a)
```

La forma empieza siendo muy abierta, pero el programa termina por llenar el círculo (en la pantalla lo vemos tras media hora).

PANTALLA DEL BALÓN



Con este tipo de programa, la resolución gráfica no consigue demasiado detalle, ya que cada curva se ha dibujado como una serie de puntos.

RELOJ DEL ORDENADOR

Las demoras de tiempo están entre los mandatos más usados en programas de ordenador. Uno de los problemas de trabajar con un micro es que a menudo ejecutan los programas demasiado deprisa. Con las demoras se deceleran útilmente los programas; por ejemplo se mantiene un texto en pantalla el tiempo necesario para poderlo leer, antes de que desaparezca, o se reduce la velocidad de una secuencia de imágenes excesivamente rápida en un programa de animación.

La manera más sencilla de programar una retención es la sentencia PAUSE, que tiene la forma de la siguiente línea de programa:

```
100 PAUSE 50
```

donde el número después de PAUSE representa el tiempo de demora en quincuagésimos de segundo. Es sencillo procesar un programa con PAUSE y ver si la demora especificada es suficiente. Si no, se especifica y prueba un número mayor después de PAUSE. Es aproximado pero suficiente; sin embargo si Vd. quiere cronometrar un suceso en un programa puede usar el reloj incorporado en el propio Spectrum.

Uso del reloj interno

Algunos ordenadores llevan un reloj que marcha independiente de lo que estén haciendo los programas. El Spectrum no tiene un mandato o función que le permita usar un sistema de tiempo con una simple sentencia de programa.

Sin embargo, el Spectrum, como el resto de los ordenadores personales lleva un reloj interno que sincroniza sus actividades, con un dispositivo contador. Tres "direcciones", o franjas, de su memoria cuentan el número de imágenes de televisión (cuadros) producidos desde que se encendió el ordenador. La primera dirección (numerada 23672) cuenta hasta un máximo de 255; después se restaura a 0 y los contenidos de la siguiente dirección (23673) se aumentan en 1. Cuando se alcanza 255, se vuelve a poner a 0 y los contenidos de la tercera (23674) se aumentan en 1. Como en Europa se producen 50 cuadros por segundo (60 en la televisión de Estados Unidos), la primera dirección cuenta en quincuagésimos de segundo; la segunda en intervalos de 5,12 seg. y la tercera en intervalos de 1310,72 seg.

Si la precisión del reloj interno es sólo de un quincuagésimo de seg. Vd. puede preguntarse qué ventaja tiene usarlo en lugar de la sentencia PAUSE, de aproximadamente la misma exactitud. La razón es que como PAUSE, sentencia BASIC, es parte de un programa, éste es incapaz de hacer nada durante la demora. Nunca puede dar más de una interrupción temporal al programa.

En cambio, el contador trabaja de modo distinto, guardando la cuenta de lo que hace el programa —con dos excepciones. Si hace que el ordenador suene (BEEP) o usa cualquier periférico conectado al ordenador —un magnetófono o una impresora, por ejemplo— el contador para, mientras Vd. lo use y después reanuda su cuenta. Así, si quiere usar el contador como reloj, aquél perdería tiempo durante estas operaciones.

Cronometrado con el contador

Para ver trabajar al contador, teclee las dos líneas siguientes:

```
10 PRINT PEEK 23672
20 GOTO 10
```

La pantalla debe presentar números crecientes (hasta 255), hasta que aparezca la petición "scroll?" en la pantalla, como se indica más abajo:

PANTALLA DEL CONTADOR

El mandato PEEK mira en la dirección especificada qué número está almacenado allí. Es la pareja de POKE, que pone un número en una dirección. PEEK 23672 significa "el contenido del" número de dirección 23672. Vea que los números visualizados no son consecutivos; hay una diferencia de tres o cuatro entre cada par. Es debido a que las propias sentencias PRINT y GOTO requieren tiempo para realizarse y el programa no es capaz de "cazar" cada simple "tick" de quincuagésimo de segundo del contador. Puede ser un problema cuando quiera usar el contador para contar en pequeños intervalos de tiempo, como en el programa siguiente. Cuando lo procesa, debe sonar un "BEEP" cada segundo. La variable t se fija igual al contenido de 23672:

"RELOJ" CONTADOR DE CUADROS

```
  10 LET t=PEEK 23672
  20 IF PEEK 23672-t=50 THEN BEE
P 0.05,20: GO TO 10
  30 GO TO 20


0 OK, 0:1
```

Cuando la diferencia entre el PEEK 23672 y el t fijado alcanza 50, ha transcurrido un segundo y debe sonar el "BEEP".

Cuando procese este programa, descubrirá que produce apenas un par de "BEEP". La razón es que PEEK 23672–t es poco probable que sea igual a 50 en el momento que se realiza la línea 20. También, la sentencia BEEP detiene el contador de cuadros durante un quincuagésimo de segundo cada vez que suena. Esta clase de programa puede usarse como un temporizador, pero el intervalo de tiempo (entre dos ticks) debe ser suficientemente grande comparado con el tiempo requerido para realizar las sentencias del programa.

El siguiente programa mide un intervalo de 5 segundos, aunque no es del todo exacto. La segunda dirección cuenta cada vez que se llena la primera —es decir, cada 256 quincuagésimos de segundo, o sea 5,12 seg. Esto es adecuado para un temporizador que sólo redondea al minuto o medio minuto. La línea 10 restaura la dirección a 0 cada vez que se procesa el programa:

TEMPORIZADOR DE 5 SEGUNDOS

```
  10 POKE 23673,0
  20 LET t=PEEK 23673
  30 IF PEEK 23673>t THEN PRINT
"tick"
  40 GO TO 30


0 OK, 0:1
```

Uso del tiempo en programas

Como mirar el contador de cuadros requiere una serie de cálculos, se puede emplear una función FNt() para representar el número de segundos transcurridos desde que se encendió el ordenador. Se establece una variable T igual a éste y, después en el programa, cuando FNt() excede T, los segundos se aumentan en 1. Después T se restaura a FNt() para la cuenta siguiente. Un programa puede también incorporar pruebas para aumentar los minutos y horas visualizadas.

Este tiempo, relativamente exacto, es de más valor en los programas. Aquí hay una forma de usarla, en un programa que controla la velocidad de sus reacciones:

TEMPORARIZADOR DE REACCIONES

```
  10 DEF FN t()=65536*PEEK 23674
+256*PEEK 23673+PEEK 23672
  20 PRINT AT 5,6;"Time your rea
ctions"
  30 PRINT AT 10,10;"Press a key
"
  40 PRINT AT 12,4;"when you hea
r the tone"
  50 PAUSE 100+4*(INT 100*RND)
  60 LET T=FN t(): BEEP 0.1,25
  70 PRINT AT 16,6;"Your time st
arts now"
  80 IF INKEY$="" THEN GO TO 80
  90 CLS
 100 PRINT AT 8,12;"You took";AT
 10,13;(FN t()-T)/50;AT 12,12;"s
econds"
 110 PAUSE 150
 120 CLS : GO TO 20


0 OK, 0:1
```

```
Time your reactions


Press a key
when you hear the tone

Your time starts now
```

La función FNt() se define en la línea 10. Ve (PEEK) las tres franjas de contadores de cuadros. Se omite la división final por 50; así la función cuenta en quincuagésimos de segundo en vez de en segundos. La línea 50 produce una demora, de al menos 2 segundos, antes de iniciarse el período medido. La línea 80 espera a que se pulse una tecla antes de borrar la pantalla (línea 90) y después calcula y visualiza el tiempo de reacción.

USO DE MATRICES

Una matriz es una forma de almacenar hechos y/o valores en la memoria del ordenador en forma de tabla, para que se pueda localizar un elemento sin tener que "recorrer" todos los anteriores. Cada elemento se especifica por uno o más números. En la siguiente matriz cada elemento se identifica inequívocamente por un par de coordenadas:

	1	2	3	4	5	6
1	PACO	ALBA	JUAN	ROSA	JOSE	LOLA
2	100	250	840	125	223	691

A esta matriz se le llama de 6×2 (6 columnas y 2 filas). El elemento (2,2) es 250; el (1,3) es JUAN, etc. Esta matriz, por necesitar dos coordenadas de identificación, se llama bi-dimensional. Si sólo hubiera tenido una fila sería uni-dimensional. La palabra clave DIM del BASIC se usa para decir al ordenador cuál debe ser el tamaño de la matriz.

Puede usarse una matriz unidimensional para almacenar una lista de números muy usados:

MATRIZ UNI-DIMENSIONAL

```
  10 DATA "January","February","
March","April","May","June","Jul
y","August","September","October
","November","December"
  20 DIM m$(12,9)
  30 FOR n=1 TO 12: READ m$(n):
NEXT n
  40 CLS
  50 FOR n=1 TO 12
  60 PRINT AT 4+n,12;m$(n)
  70 NEXT n


0 OK, 0:1
```

Aquí la línea 20 le dice al ordenador que la matriz m tiene 12×9 entradas (9 es el máximo de letras de cualquiera de los 12 meses en inglés). El programa visualiza la lista de los meses del año dada en la línea 10. Aunque hay formas más sencillas de hacerlo, después en un programa, puede querer enlazar un mes con otra información o el resultado de un cálculo. Usando este listado, puede "extraer" cualquier mes usando m$(n), siendo n el número del mes. Cuando se procesa el programa, la pantalla aparecerá así: una tabla de meses lista para más información:

PANTALLA DE MATRIZ UNI-DIMENSIONAL

```
          January
          February
          March
          April
          May
          June
          July
          August
          September
          October
          November
          December


0 OK, 70:1
```

Cómo añadir una dimensión

Ahora elaboremos el programa de matriz de calendario para hacer algo útil. Añada una segunda matriz, numérica, de manera que pueda listar algunos valores frente a cada mes. El siguiente programa usa la matriz para representar la pluviometría mensual.

Ahora la tabla tiene dos cabeceras. No necesita visualizar todos los elementos de la matriz de cadenas —m$(n)— antes de moverla para seleccionar la matriz numérica —r(n). La línea 80 toma un elemento de cada matriz. Como estos se han de visualizar en filas consecutivas de la pantalla, pueden identificarse fácilmente relacionándolos al número de fila. Para cada valor de n, m$(n) y r(n) son visualizados para diferentes números de columna de la fila (5+n):

MATRIZ BI-DIMENSIONAL

```
  10 DATA "January","February","
March","April","May","June","Jul
y","August","September","October
","November","December"
  20 DATA 2.5,1.0,2.5,7.5,9.5,4.
5,3.5,4.0,4.5,4.5,4.0,2.5
  30 DIM m$(12,9): DIM r(12)
  40 FOR n=1 TO 12: READ m$(n):
NEXT n
  50 FOR n=1 TO 12: READ r(n): N
EXT n
  60 PRINT AT 4,6;"MONTH";AT 4,1
8;"RAINFALL (cms)"
  70 FOR n=1 TO 12
  80 PRINT AT 5+n,6;m$(n);AT 5+n
,20;r(n)
  90 NEXT n


0 OK, 0:1
```

PANTALLA DE MATRIZ BI-DIMENSIONAL

```
        MONTH          RAINFALL (cms)

        January          2.5
        February         1
        March            2.5
        April            7.5
        May              9.5
        June             4.5
        July             3.5
        August           4
        September        4.5
        October          4.5
        November         4
        December         2.5


0 OK, 90:1
```

Cómo escribir tablas con matrices

Vd. puede ahora escribir una tabla más ambiciosa:

PROGRAMA DE TABLA DE IMPUESTOS

```
  10 DATA 1.98,2.40,5.60,1.05,4.
35,2.99,1.92,7.20,5.45
  20 DATA 20,19,11,45,15,15,24,4
,6,44,84,108,164,204
  30 LET t=15
  40 DIM q(9,2)
  50 FOR c=1 TO 2
  60 FOR r=1 TO 9
  70 READ q(r,c)
  80 NEXT r
  90 NEXT c
 100 PLOT 4,12: DRAW 248,0: DRAW
 0,160: DRAW -248,0: DRAW 0,-160
 110 FOR n=12 TO 156 STEP 16
 120 PLOT 4,n: DRAW 248,0
 130 NEXT n
 140 FOR n=1 TO 5
 150 READ x
 160 PLOT x,12: DRAW 0,160
 170 NEXT n
 180 PRINT AT 1,1;"ITEM";AT 1,6;
"COST";AT 1,11;"No";AT 1,14;"SUB
scroll?
```

```
 190 FOR n=1 TO 9
 200 PRINT AT 2*n+1,2;n;AT 2*n+1
,6;q(n,1);AT 2*n+1,11;q(n,2)
 210 PRINT AT 2*n+1,14;q(n,1)*q(
n,2);AT 2*n+1,21;(INT (((q(n,1)*
q(n,2)*t/100)+0.005)*100))/100;A
T 2*n+1,26;(INT (((q(n,1)*q(n,2)
*(1+t/100))+0.005)*100))/100
 220 NEXT n
 230 PAUSE 150
 240 PRINT AT 21,7;"Try a new ta
x rate": INPUT t
 250 PRINT AT 21,7;"
     "
 260 GO TO 180


0 OK, 0:1
```

En este programa de planificación financiera, las columnas están interrelacionadas, y Vd. puede cambiar algo de la información visualizada, si es necesario.

La línea 10 contiene los datos (DATA) para la primera parte de la matriz (serie de precios), y la línea 20, los de la segunda parte (serie de cantidades). La línea 20 también contiene algunas coordenadas que se usarán después en el programa. Las líneas 40 a 90 dimensionan la matriz 9×2 y "leen" sus datos. Las líneas 100 a 170, dibujan la red que encuadra los datos. Las coordenadas del extremo inferior de las líneas verticales de la red se almacenan en la línea 20. La línea 180 visualiza las cabeceras de columnas.

Los datos se visualizan en la red por las líneas 190 a 220. Se han de visualizar cada dos líneas desde las filas 3 a 19. El número de elemento, n (del 1 al 9) se relaciona a la fila así:

row=2*n+1

Esto aparece en las líneas 200 y 210 en las sentencias PRINT AT. Las dos últimas partidas de la línea 210 parecen especialmente complejas. Si el subtotal fue 8,25 el impuesto se calcularía así: 0.15*8.25=1.2375 —demasiados decimales; se multiplica, pues, el impuesto por 100, se toma el valor entero (eliminando todos los decimales) y después se divide por 100. Se añade 0.005 para redondear el valor final a la siguiente unidad.

Las líneas 240 a 260 le invitan a introducir un nuevo índice de impuesto. Si lo hace y pulsa ENTER, se recalculan todos los valores de la tabla que usan el índice de impuesto. Esta posibilidad de recálculo instantáneo es el principio de un tipo de programa de planificación financiera llamado "hoja de cálculo". Pueden introducirse columnas interrelacionadas de valores que representen los ingresos, costes de materiales/producción, gastos generales, etc. Entonces puede observar los efectos de cambiar uno o más de estos parámetros, ya que todos los parámetros se recalculan en toda la pantalla:

PANTALLA DE TABLA DE IMPUESTOS

ITEM	COST	No	SUB	TAX	TOTAL
1	1.98	20	39.6	3.17	42.77
2	2.4	19	45.6	3.65	49.25
3	5.6	11	61.6	4.93	66.53
4	1.05	45	47.25	3.78	51.03
5	4.35	15	65.25	5.22	70.47
6	2.99	15	44.85	3.59	48.44
7	1.92	24	46.08	3.69	49.77
8	7.2	4	28.8	2.39	31.19
9	5.45	6	32.7	2.62	35.32

Try a new tax rate

■

TRABAJO CON PALABRAS 1

Casi todos los programas considerados hasta aquí han tomado datos numéricos y ejecutado cálculos para llegar a un resultado. Pero las cadenas de caracteres se han dejado casi invariablemente en su orden original. Sin embargo ya vio Vd. en el programa de reacciones (pág. 13) que el ordenador también almacena letras en código ASCII (ver pág. 60). Como estos códigos tienen valores numéricos, el ordenador puede examinar cadenas y después reordenarlas de forma similar a los números.

Cómo reordenar números

Para ver cómo se clasifican o reordenan palabras —o cualquier cadena— es útil que sepa cómo se hace con números. Aquí hay un programa que le pide seis números y después los reordena en orden numérico. Puede ampliarlo fácilmente para tratar más de seis:

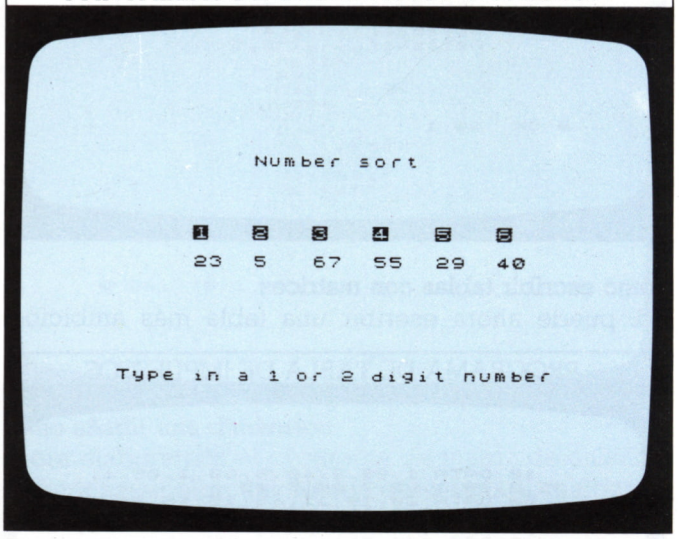
PROGRAMA DE ORDENACIÓN NUMÉRICA

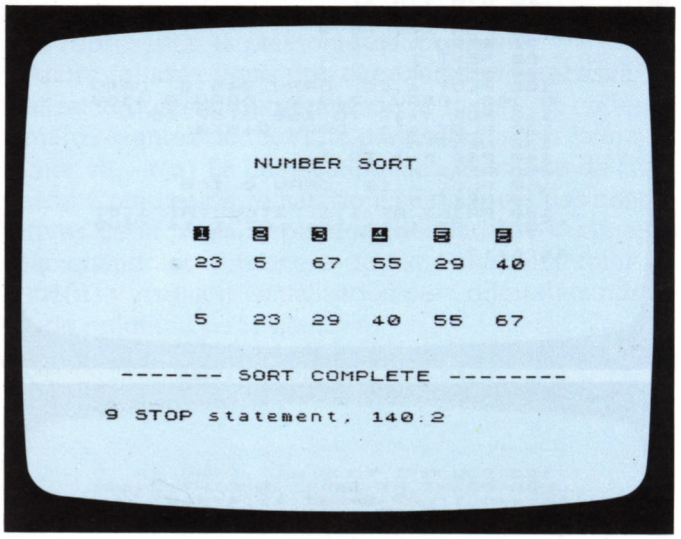
```
  10 DIM a(6)
  20 PRINT AT 5,10;"NUMBER SORT"
  30 FOR n=1 TO 6
  40 PRINT PAPER 0; INK 7;AT 10,
2+4*n;n
  50 PRINT AT 20,1;"Type in a 1
or 2 digit number"
  60 INPUT a(n): PRINT AT 12,2+4
*n;a(n)
  70 NEXT n
  80 FOR t=1 TO 5: FOR n=1 TO 5
  90 IF a(n+1)<a(n) THEN GO SUB
150
 100 NEXT n: NEXT t
 110 FOR n=1 TO 6
 120 PRINT AT 16,2+4*n;a(n)
 130 NEXT n
 140 PRINT AT 20,1;"------- SORT
COMPLETE --------": STOP
 150 LET b=a(n): LET a(n)=a(n+1)
: LET a(n+1)=b
 160 RETURN

0 OK, 0:1
```

La línea 50 le pide que teclee un número de uno o dos dígitos. Cada vez que lo hace y pulsa ENTER, se visualiza la siguiente petición (de 1 a 6) sobre la siguiente posición INPUT. Aquí es donde su número elegido se visualiza por la línea 60. Todos los números introducidos se cargan en una matriz, de forma que puedan etiquetarse para identificarlos.

Después de introducidos seis números, comienza automáticamente la ordenación en la línea 80. Primero examine el bucle n (desde la segunda sentencia de la línea 80 a la 100). Para cada valor de n, se compara a(n+1) con a(n); si a(n+1) es el menor, la línea 150 los invierte. Así, si en la primera pasada del bucle (n=1) sus dos primeros números fueron 34 y 16, se llamaría a la subrutina de inversión, ya que el segundo número es menor que el primero. Se establecería b=a(1º) (34 en este caso); a(1º)=a(2º) (16 aquí); y, finalmente, a(2º)=b, que en este bucle es 34. Este "truco" invierte los dos números. El número máximo de ordenaciones necesarias para poner la lista de números en orden correcto es cinco, y así el bucle t repite la rutina de ordenación 5 veces. Las siguientes pantallas muestran los números antes y después de ordenados:

PANTALLAS DE ORDENACIÓN NUMÉRICA





Este proceso de inversión puede usarse también para manejar cadenas. Es fácil ver cómo el ordenador compara dos números y "ve" si el segundo es menor que el primero. Lo hacemos siempre nosotros cuando comparamos precios. Pero el ordenador también puede decidir si "Londres" es menor que "Nueva York", es decir si es anterior en el alfabeto. La línea:

```
50 IF "New York"<"London" THEN GOSUB 300
```

a primera vista parece sin sentido. Pero como el ordenador almacena una cadena como Nueva York o Lon-

dres, como una serie de números, éstos pueden compararse y reordenarse. En BASIC, pues, tienen sentido comparaciones a<b o Estocolmo>París.

Redisposición en orden alfabético

Una de las aplicaciones más útiles de clasificación de cadenas es ordenarlas alfabéticamente. El siguiente programa muestra un método —trabaja con cadenas ya incorporadas en el programa, pero se puede adaptar el programa para aceptar ristras diferentes usando INPUT:

PROGRAMA DE CLASIFICACIÓN ALFABÉTICA

```
  10 DATA "Paris","Stockholm","N
ew York","London","Rome","Amster
dam"
  20 DIM a$(6,9)
  30 FOR n=1 TO 6: READ a$(n): N
EXT n
  40 PRINT AT 5,9;"Alpha sort 1"
  50 GO SUB 120
  60 FOR t=1 TO 5
  70 FOR n=1 TO 5
  80 IF a$(n+1)<a$(n) THEN GO SU
B 150
  90 NEXT n
 100 GO SUB 120
 110 NEXT t: STOP
 120 FOR n=1 TO 6
 130 PRINT AT 2*n+7,10;a$(n)
 140 PAUSE 25: NEXT n: RETURN
 150 LET b$=a$(n)
 160 LET a$(n)=a$(n+1)
 170 LET a$(n+1)=b$
 180 RETURN

0 OK, 0:1
```

La línea 20 dimensiona una matriz de cadenas usando el método de la página 22. La línea 30 lee el contenido de la línea 10 —una serie de capitales. La subrutina de las líneas 120 a 140 visualiza las ciudades en su orden original. Después, la rutina de ordenación mueve las cadenas hasta que quedan en orden alfabético. Podría añadir una pausa (PAUSE) paa que pueda ir viendo lo que sucede:

PANTALLAS DE ORDENACIÓN ALFABÉTICA

```
Alpha sort 1

 Paris
 Stockholm
 New York
 London
 Rome
 Amsterdam
```

```
Alpha sort 1

 Amsterdam
 London
 New York
 Paris
 Rome
 Stockholm

9 STOP statement, 110:2
```

La rutina de ordenación en las líneas 150 a 180 es básicamente la misma que la de ordenar números. Las variables usadas aquí son cadenas alfabéticas pero el ordenador trabaja de la misma manera que con números. La línea 110 detiene el programa.

Vd. se dará cuenta de que el almacenamiento temporal usado en ambos programas (b y b$) no necesita ser un elemento de una matriz. La variable se usa sólo una vez en cada ordenación y después ya no se necesita.

Cómo cambiar cadenas a números

El Spectrum usa una serie de palabras clave que le permiten utilizar información numérica a partir de cadenas. Además de tomarlas y comparar sus valores ASCII, el ordenador puede decidir cuál es la longitud. Para programarlo se usa la tecla LEN. La línea siguiente, por ejemplo, produce un número:

```
100 n=LEN"Spectrum"
```

Aquí n es 8, longitud de la cadena. Puede usarlo en los programas para rechazar una entrada de nombre o palabra que sea demasiado largo, o hacer que el ordenador tome una acción diferente según la longitud de alguna cadena. Puede sumar LEN juntos para hallar la longitud de un número de nombres o de cualquier parte de texto.

El Spectrum además tiene una palabra clave que trabaja con cadenas que son números. VAL convierte un número-cadena que contiene un cálculo, en un número. El número producido es el resultado del cálculo. Por ejemplo:

```
150 LET a$="2*34.5*0.3"
160 PRINT a$;"=";VAL a$
```

visualiza tanto el cálculo como su resultado, lo que es útil en programación. Puede usar este mandato para valorar una cadena y después pasar este valor a otra parte del programa.

TRABAJO CON PALABRAS 2

Hasta ahora, ha tratado las cadenas —o palabras que se reúnen en cadenas— como unidades indivisibles. Algún programa ha sumado cadenas, pero ninguno ha "mirado dentro" de las comillas que encierran cada cadena para trabajar con los caracteres interiores. Con el Spectrum puede reagrupar los caracteres de una cadena de formas diferentes, es decir que puede programar al ordenador para extraer parte de una palabra o grupo de palabras y examinarlas —proceso que puede ser muy útil.

La mayoría de los ordenadores que trabajan con BASIC usan los mandatos LEFT$, RIGHT$, MID$, etc. para manejar cadenas; sirven para escoger el carácter primero, último o intermedio de una cadena, respectivamente. Aunque estos mandatos no están en el teclado del Spectrum el ordenador puede hacer todo lo que hacen estos comandos, si Vd. sabe programarlo.

Cómo cortar palabras

Vd. puede hacer que el Spectrum rompa una palabra cortando la cadena. Es muy sencillo. Para ver cómo se puede hacer teclee el listado de la pantalla siguiente:

PROGRAMA CORTADOR DE CADENAS

```
  10 LET a$="Slicing words with
the Spectrum"
  20 CLS
  30 FOR n=1 TO 31
  40 PRINT a$( TO n)
  50 NEXT n


0 OK, 0:1
```

Aquí la técnica está en la línea 40, donde aparece un mandato TO como parte de una variable en cadena. Para cada valor de n, la línea 40 visualiza una cadena de longitud n. Así la primera línea contiene la cadena "S", la segunda "Sl" y así sucesivamente, hasta que n iguala la longitud establecida. Con este programa puede usar cualquier tipo de cadena. Es mejor que el valor de n no supere los 32 caracteres de una línea de pantalla. Usando una cadena diferente, asegúrese de que el valor máximo de n en la línea 30 no sea mayor que la longitud de su cadena:

PANTALLA CORTADORA DE CADENAS

```
S
Sl
Sli
Slic
Slici
Slicin
Slicing
Slicing
Slicing w
Slicing wo
Slicing wor
Slicing word
Slicing words
Slicing words
Slicing words w
Slicing words wi
Slicing words wit
Slicing words with
Slicing words with
Slicing words with t
Slicing words with th
Slicing words with the

scroll?
```

Puede usar esta técnica para elegir cadenas que empiecen por la misma letra o palabra, y después quizás visualizarlas en una serie de listas.

El efecto inverso es igual de fácil. Añada las líneas siguientes al programa, procéselo y vea lo que pasa:

```
60 FOR n=1 TO 31
70 PRINT a$(TO 32-n)
80 NEXT n
```

Ahora, como n aumenta de 1 a 31 en cada bucle, la longitud de la cadena visualizada decrece desde 31 caracteres a 1.

Cómo extraer partes de una frase

Ahora puede explorar esta técnica. Teclee y procese el listado de la pantalla siguiente:

PROGRAMA CORTADOR SELECTIVO DE CADENAS

```
  10 LET a$="DK Screen Shot Seri
es"
  20 CLS
  30 PRINT AT 5,9;"String Choppe
r"
  40 PRINT AT 7,6;a$
  50 PRINT AT 10,6;a$( TO 2)
  60 PRINT AT 12,6;a$(4 TO 14)
  70 PRINT AT 14,5;a$(15 TO )




0 OK, 0:1
```

No está limitado a tratar los n primeros caracteres de una cadena, puede tomar cualquier grupo de caracteres consecutivos desde una palabra a una frase. En este programa la línea 50 trabaja de la misma manera que la 40 del programa cortador. La línea 60 forma una cadena desde los caracteres 4 a 14 del interior de a$. Finalmente, la línea 70 forma una tercera cadena desde el carácter decimoquinto al final de a$. Aunque estas "subcadenas" se forman como parte de a$, la misma a$ está aún intacta. Esta técnica permite tomar un grupo de palabras y elegir cualquiera de ellas para usarla sola en un programa.

Juegos de palabras con mandatos de cadenas

El siguiente programa muestra cómo usar estos métodos de manejo de palabras en un juego. Es un acertijo de palabras con ordenador, en el que un jugador introduce una palabra y otro tiene que acertarla; el ordenador visualiza las letras que el usuario ha acertado en sus posiciones correctas en la palabra:

PROGRAMA DE LA HORCA

```
  10 BORDER 0: CLS : PRINT AT 1,
12;"HANGMAN"
  20 PRINT AT 10,2;"Ask a friend
 to type a word"
  30 PRINT AT 12,3;"or phrase fo
r you to guess"
  40 PRINT AT 18,4;"DON'T LOOK A
T THE SCREEN"
  50 PRINT AT 20,0;"Press ENTER
when you're finished"
  60 INPUT a$
  70 CLS : LET l=LEN a$: LET s=0
  80 FOR n=1 TO l
  90 IF a$(n)=" " THEN PRINT AT
11,(32-l)/2+n;"■": NEXT n
 100 PRINT AT 11,(32-l)/2+n;"-":
 NEXT n
 110 PRINT AT 2,12;"HANGMAN";AT
5,6;"-:letter  ■:space"
 120 PRINT AT 18,6;"  Try a le
tter    ";AT 19,0;"Press 1 to gu
ess the whole thing"

scroll?
```

```
 130 INPUT t$
 140 IF t$="2" THEN STOP
 150 IF t$="1" THEN GO TO 210
 160 LET s=s+1: PRINT AT 16,12;"
tries=";s
 170 FOR n=1 TO l
 180 IF t$=a$(n) THEN PRINT AT 1
1,(32-l)/2+n;t$
 190 NEXT n
 200 GO TO 130
 210 PRINT AT 18,6;"Try the whol
e thing": INPUT t$
 220 IF t$=a$ THEN CLS : PRINT A
T 11,12;"CORRECT";AT 13,12;"scor
e=";s+1: STOP
 230 GO TO 120



0 OK, 0:1
```

Las líneas 10 a 50 visualizan la estructura del título. Cuando un amigo ha tecleado la cadena que se ha de adivinar, la línea 70 calcula la longitud con el mandato LEN, y pone la cuenta, s, a cero.

Ahora el programa ha de visualizar símbolos sobre la pantalla para representar las letras de la cadena. Cuando se aciertan letras éstas reemplazan a los símbolos. También, para permitir jugar con frases y no sólo con palabras, se muestra dónde están los espacios que separan palabras. La línea 100 visualiza guiones que representan letras. La línea 90 visualiza cuadrados para representar los espacios de las cadenas. En las líneas 90 y 100, (32-l)/2+n establece dónde deben visualizarse los caracteres de la cadena para que aparezcan en el centro de la pantalla, no descentrados (un efecto similar se incorpora en los programas de proceso de textos).

Si Vd. quiere adivinar la palabra o frase entera, en vez de teclear letras individuales (esto puede hacerse en cualquier punto del juego), pulse 1. El programa salta a la línea 210. La palabra o frase que se tecleó en (t$) se compara con la cadena almacenada (a$). Después se visualiza un "correcto" o si la respuesta es errónea, el programa vuelve a la entrada de letra única. Cuando se prueba con letras, las líneas 170 a 190 las comparan una a una con cada carácter de la cadena almacenada. Si concuerda, se visualiza en la posición adecuada. Este es un ejemplo de la pantalla que se debe ver durante el juego:

PANTALLA DE LA HORCA

```
HANGMAN

-:letter    ■:space


--e-tr--


tries=3
Try a letter
Press 1 to guess the whole thing
```

Puede limitar el número de respuestas añadiendo los mandatos:

```
IF s>n THEN STOP
```

después de las sentencias donde se calcula la cuenta, s. Si Vd. tuviera un error en el programa, puede ser difícil parar usando la tecla BREAK en CAPS SHIFT, así para facilitarlo añada otra línea:

```
145 IF t$="3" THEN STOP
```

Para parar el programa en cualquier punto, pulse 3.

COMO ENCONTRAR DATOS

Vd. puede usar su Spectrum para almacenar información como archivo electrónico. Sin embargo, exigiría tiempo ver todo el contenido de un archivo grande cada vez que quiera consultar una partida. Incluso tendría que explorar la pantalla para encontrar la información que busca como en los archivos de papeles. Un buen programa le permite escoger información, de forma que la búsqueda la haga el ordenador y no Vd.

Cómo programar una búsqueda en serie

El siguiente programa usa un método simple pero eficaz de localizar una partida en una larga lista de datos. Se llama "en serie" porque busca el dato en una serie de etapas. Toma la cadena (T$) que Vd. teclea y la compara una a una con cada cadena en las sentencias DATA del programa, hasta que la cadena es igual a una de las almacenadas:

PROGRAMA DE BÚSQUEDA EN SERIE

```
  10 DATA "Anne","93-446 2039","
Dentist","920 4263","Doctor","29
2 3042","Elizabeth","021-111 488
4","Fred","032-244 9220"
  20 DATA "Jim","021-437 2466","
John","01-444 9000","Mac","01-60
0 0024","Pat","429866","Police",
"866529","Richard","01-628 1900"
  30 DATA "Ross","816594","Schoo
l","986881","Sheila","01-626 824
4","Taxi","629 4306","Vet","7044
6","Wendy","260 8377"
  40 BORDER 0: PAPER 7: INK 0: C
LS
  50 PRINT AT 5,9;"Serial search
"
  60 PRINT AT 10,6;"Enter name:-
----------"
  70 PRINT AT 20,2;"Press ENTER
to start search"
  80 INPUT T$: PRINT AT 10,17;T$
  90 LET t=1

scroll?
```

```
 100 FOR n=1 TO 17
 110 READ N$,P$
 120 IF N$=T$ THEN GO SUB 190: S
TOP
 130 LET t=t+1
 140 NEXT n
 150 PRINT AT 20,2;"------Search
complete-------"
 160 BEEP .2,18
 170 PRINT AT 10,6;"---Name not
found-----"
 180 STOP
 190 PRINT AT 20,2;"------Search
complete------"
 200 BEEP .2,25
 210 PRINT AT 12,13;P$
 220 RETURN



0 OK, 0:1
```

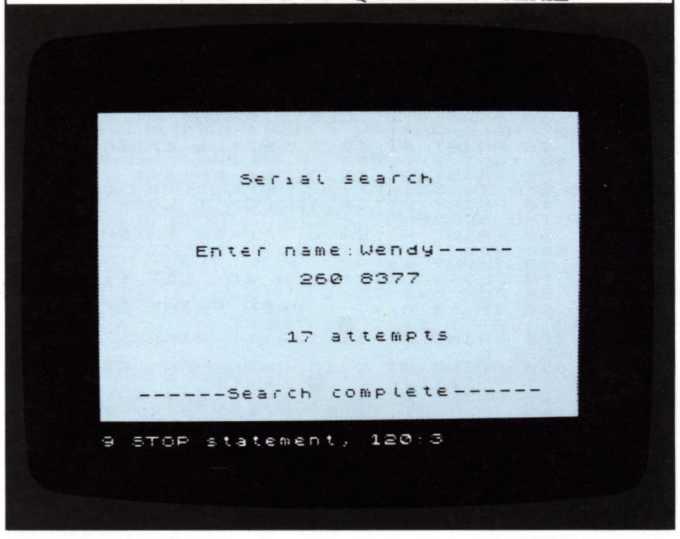
Las líneas 10 a 30 guardan una lista de nombres y números telefónicos. Las líneas 100 a 140 "leen" varias veces un nombre (N$) y número (P$), de las sentencias DATA (los números se tratan como cadenas). Así, hasta que el nombre almacenado sea igual que la cadena. La línea 210 visualiza después el número de teléfono.

Si el ordenador no puede igualar la cadena con ninguna de las almacenadas, la línea 170 anuncia que no la ha encontrado. Si ahora añade la línea siguiente Vd. podrá contar el número de búsquedas hechas hasta conseguirlo:

```
215 PRINT AT 16,12;t;"attempts"
```

Ahora intente procesar el programa para hallar el último número de teléfono en la lista:

PANTALLA DE BÚSQUEDA EN SERIE



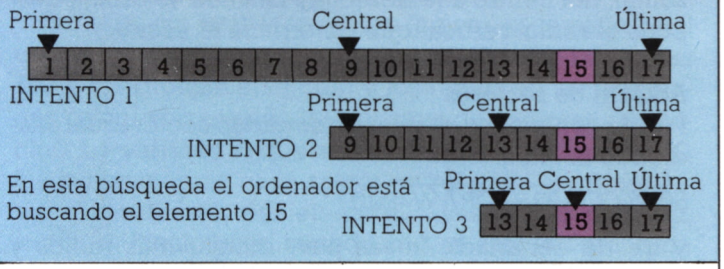
Verá que el programa emplea 17 búsquedas y una fracción de segundo para hallar el último nombre de la lista. Es aceptable en un programa corto como este, con pocos datos almacenados. Pero para un programa de búsqueda en serie de unos 200 nombres, u otros elementos, la demora causada al mirar cada elemento en la búsqueda sería notable. El programa es sencillo, pero lento.

Cómo acelerar la búsqueda de datos

Otro método de búsqueda puede acelerar el proceso. Se basa en almacenar los datos en orden numérico o alfabético. Los datos almacenados en el programa se dividen en dos repetidamente y la primera mitad se "lee" y compara con la cadena en prueba. Esto da tres posibilidades. Primera, las dos cadenas pueden ser idénticas —éxito al primer intento. Segunda, la cadena puede resultar posterior en orden alfabético o numérico que la encontrada. Tercera, puede resultar anterior. Si la igualación no tiene éxito, la mitad del dato en que

cae la cadena de prueba se vuelve a subdividir en dos y se busca de igual forma hasta que se igualen.

ETAPAS EN UNA BÚSQUEDA DISECCIONADA



Este es un programa que efectúa la "búsqueda biseccionada" de un banco de datos:

```
  10 DATA "Anne","Dentist","Doct
or","Elizabeth","Fred","Jim","Jo
hn","Mac","Mark","Pat","Police",
"Richard","Ross","Sheila","Taxi"
,"Vet","Wendy","ABCDE"
  20 DATA "93-446 2039","920 426
3","292 3042","021-111 4884","03
2-244 9220","021-437 2466","01-4
44 9000","01-600 0024","946882",
"429866","866529","01-628 1900",
"816594","01-626 8244","629 4306
","70446","260 8377"
  30 BORDER 0: PAPER 7: INK 0: C
LS
  40 DIM N$(18,9): DIM P$(17,11)
  50 FOR n=1 TO 18: READ N$(n):
NEXT n
  60 FOR n=1 TO 17: READ P$(n):
NEXT n
  70 LET f=1: LET l=18
  80 PRINT AT 5,8;"Bisection sea
rch"

scroll?
```

```
  90 PRINT AT 10,6;"Enter name:-
---------"
 100 PRINT AT 20,2;"Press ENTER
to start search"
 110 INPUT T$: PRINT AT 10,17;T$
 120 LET t=1
 130 LET x=INT ((f+l)/2)
 140 IF N$(x)=T$ THEN GO TO 190
 150 IF N$(x)<T$ THEN LET f=x
 160 IF N$(x)>T$ THEN LET l=x
 170 IF x=INT ((f+l)/2) THEN GO
TO 220
 180 LET t=t+1: GO TO 130
 190 PRINT AT 20,2;"------Search
complete------": BEEP 0.2,20
 200 PRINT AT 12,13;P$(x)
 210 STOP
 220 PRINT AT 20,2;"------Search
complete------": BEEP 0.2,10
 230 PRINT AT 10,6;"---Name not
found----"
 240 STOP

0 OK, 0:1
```

El dato se ha reorganizado en dos listados separados —uno de nombres y un segundo de números de teléfono. Para localizar el elemento en estas sentencias DATA sin tener que "leer" todos, debe numerarse o etiquetarse de alguna forma. Lo hace la línea 40 creando dos listas numeradas de datos o matrices. Las líneas 50 y 60 "leen" los elementos en las dos matrices. La línea 130 fija las condiciones para la primera búsqueda. El primer elemento a dividir, f, es uno y el último, l, es 18. Si no se hace igualación en la línea 140, las líneas 150 y 160 fijan los nuevos valores de f y l para la búsqueda siguiente.

Si procesa el programa de la forma que se indica aquí, contestará "Name not found" a cada cadena de prueba, debido a que la línea 40 dice que cada cadena (N$) tiene 9 caracteres de longitud y cada cadena (P$) 11. Un nombre como "Wendy" se almacena, pues, como "Wendy " que, en cuanto al ordenador, no es igual a la cadena de prueba. Para resolver esto añada:

```
111 LET T=LEN T$
112 IF T<>9 THEN LET T$=T$+" ": GOTO 111
205 PRINT AT 16,12;t;"attempts"
```

Así se suman espacios a la cadena de prueba hasta que tiene 9 caracteres de longitud. Ahora el programa se procesa y visualiza el número de intentos para encontrar la igualación.

PANTALLA DE BÚSQUEDA BISECCIONADA

```
Bisection search

Enter name:Wendy-----
260 8377

5 attempts

------Search complete------

0 STOP statement, 210:1
```

La última partida en la línea 10 no es errata de imprenta. Debido a que el programa divide los datos en mitades, el último nombre (WENDY en este caso) nunca se localizaría. El valor máximo de x es uno menos que el número de datos: Para soslayar esto, se añade la falsa partida ABCDE, de forma que se pueda buscar el dato con éxito.

Usando este método, el último nombre puede encontrarse tras sólo 5 intentos. Para ahorrar tiempo, el número P$ sólo se "lee" cuando se igualó el nombre. Este ahorro hace esta técnica mucho más adecuada si se quiere buscar en largas listas de datos. Podría también usar programas como los de las págs. 24-25 para ordenar los datos antes de usarlos en el programa de búsqueda. Combinando los programas tendría un accesible banco de datos.

GRAFICOS DE SECTORES

Los gráficos de ordenador son valiosísimos para visualizar información de una forma que pueda comprenderse de un vistazo. El de sectores es uno de los más sencillos de comprender y uno de los más útiles para mostrar la relación entre una cantidad parcial y la total que la comprende.

Dibujo de un gráfico fijo de sectores

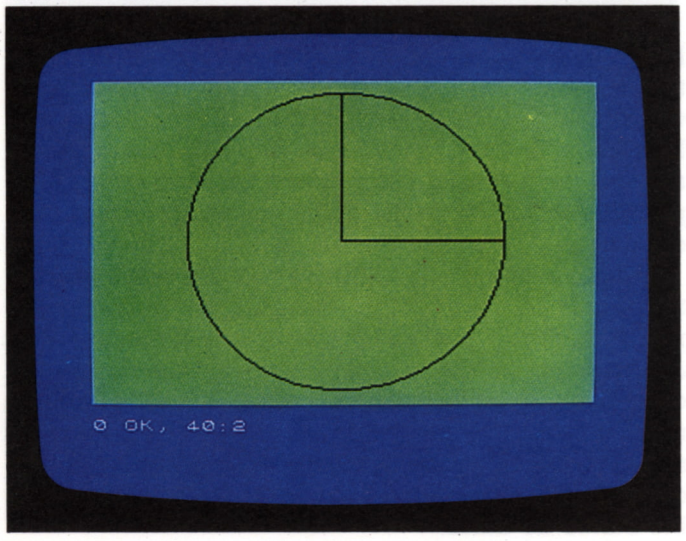
Para obtener un gráfico de sectores, lo primero es dibujar un círculo, después se marcan los bordes, (radios de los sectores). Una vez dibujado el primer radio, los demás pueden dibujarse referidos al primero. Es como cortar una tarta; no importa dónde se empieza, pero una vez iniciado, el tamaño de cada trozo está determinado por el ángulo que el sector hace con el anterior.

En el siguiente programa, se dibuja un sector de un ángulo recto en un círculo:

GRÁFICO DE SECTOR ÚNICO

```
 10 BORDER 1: PAPER 4: INK 0: C
LS
 20 CIRCLE 128,88,80
 30 PLOT 128,88: DRAW 80,0
 40 PLOT 128,88: DRAW 0,80
```

```
0 OK, 0:1
```



El programa construye el círculo en el centro de la pantalla (128,88). La línea 30 dibuja el primer radio (horizontal, del centro a la derecha). La línea 40 dibuja después el radio vertical que determina el sector.

Adición de sectores

Podría continuar y dibujar un segundo sector igual añadiendo la línea:

```
50 PLOT 128,88:DRAW -80,0
```

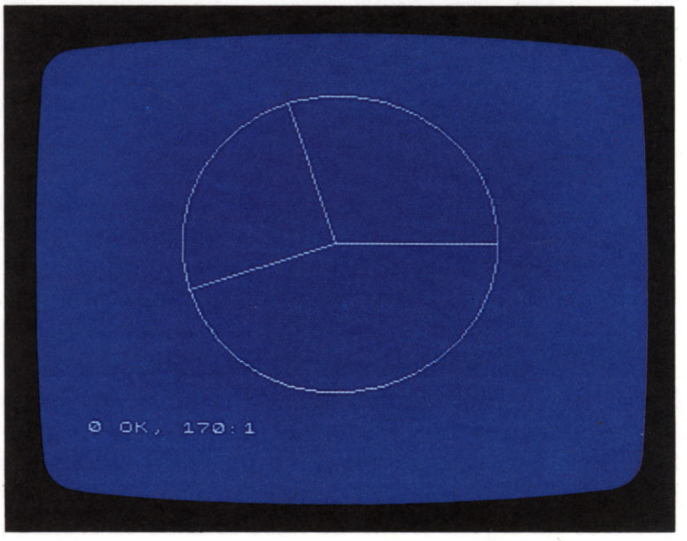
y desde ahí añadir líneas para dibujar más radios y construir un gráfico de varios sectores. Pero esta clase de programa no es muy usual ya que hay que calcular las posiciones de todos los radios.

Sin embargo, el principio básico del primer programa puede usarse para escribir otro que calcule la posición de todos los radios:

GRÁFICO VARIABLE

```
 10 DIM s(3): LET s=0
 20 PRINT AT 5,12;"PIE MAKER"
 30 PRINT AT 6,12;"**********"
 40 PRINT AT 8,7;"Total pie siz
e?"
 50 INPUT t: PRINT AT 8,26;t
 60 FOR n=1 TO 3
 70 PRINT AT 3*n+9,7;"What size
 slice ";n;"?"
 80 INPUT s(n): PRINT AT 3*n+9,
26;s(n)
 90 NEXT n
100 BORDER 1: PAPER 1: INK 7: C
LS
110 CIRCLE 128,88,80
120 PLOT 128,88: DRAW 80,0
130 FOR n=1 TO 3
140 LET s=s+s(n)
150 PLOT 128,88
160 DRAW 80*COS (s*2*PI/t),80*S
IN (s*2*PI/t)
170 NEXT n
```

```
0 OK, 0:1
```



Las líneas 20 a 90 establecen el título y el cuadro de entrada. La línea 10 dice al ordenador que el programa usará una matriz uni-dimensional llamada s que tendrá tres elementos. Cada uno de estos elementos serán del tamaño de uno de los tres sectores que el bucle de las líneas 60 a 90 le pide a Vd. que teclee.

Primero, sin embargo, debe teclear el tamaño total de sector en respuesta a las líneas 40 y 50. Las líneas 100 a 170 dibujan el círculo y los radios en las posiciones calculadas usando los tamaños de sectores que tecleó. La variable s, que se puso a cero en la línea 10, fija la posición de cada sector en relación al primer radio. Cuando n=1, por ejemplo, s=s+s(1). Cuando n=2, s=s+s(2). Finalmente, cuando n=3, a=s+s(3). Para cada valor de s se dibuja una línea desde el centro al punto calculado usando COS y SIN en la línea 160.

Puede preguntarse por qué esto puede dibujar líneas que terminan a la izquierda o debajo del centro. El COS de ángulos entre PI/2 y 3*PI/2 radianes (90 y 270 grados) es negativo. Así si COS(s*2PI/t)=-1, el radio se dibuja a un punto de abscisa x=128+(-80)=48.

Etiquetado de los sectores

El programa anterior funciona bien, pero nada identifica cuál es cada sector; se necesita buena memoria para saber qué sector representa cada una de las cantidades que tecleó. Para esto se puede añadir una rutina para etiquetar los sectores:

RUTINA DE ETIQUETADO

```
 115 PRINT AT 0,0;"Total=";t
 121 DIM x(4): DIM y(4)
 122 LET x(1)=50: LET y(1)=0
 170 LET x(n+1)=80*COS (s*2*PI/t
)
 180 LET y(n+1)=80*SIN (s*2*PI/t
)
 190 LET x=(x(n)+x(n+1))/2
 200 LET c=16+INT (x/8+1)
 210 LET y=(y(n)+y(n+1))/2
 220 LET r=11-INT (y/8+1)
 230 PRINT PAPER n; INK 7;AT r,c
;n
 240 PRINT PAPER n; INK 7;AT n,1
;n;"-";s(n)
 250 NEXT n


0 OK, 0:1
```

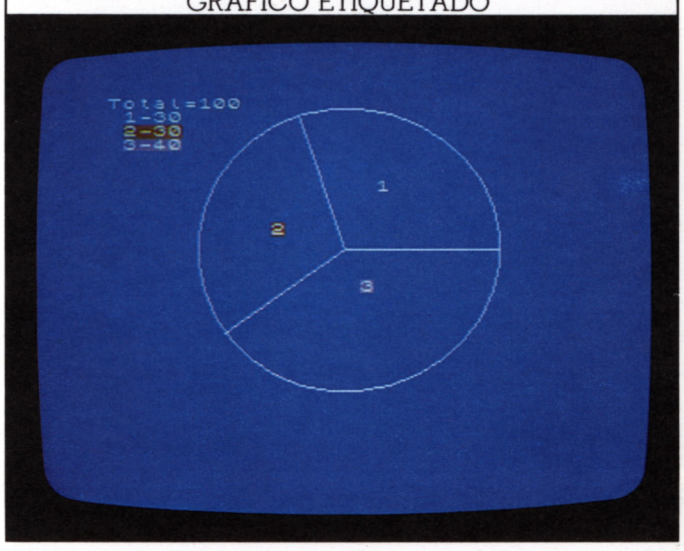
La línea 121 establece dos matrices, x e y, que se usan para almacenar las coordenadas de los extremos de radios dibujados sobre el círculo. Hay cuatro pares, no tres, el del primer radio y los de los otros tres. Las coordenadas del primer radio son conocidas (50,0) y se establecen en la línea 122.

Para cada valor de n, las coordenadas del extremo de estos radios se extraen de las matrices para x(n+1), y(n+1).

El programa usa estas coordenadas para visualizar una etiqueta sobre la pantalla. La posición de la etiqueta se da en dos coordenadas x e y. Estas coordenadas son la mitad de la distancia entre x(n) y x(n+1) respectivamente. Las dos coordenadas gráficas se traducen en números de fila y columna por las líneas 200 y 220.

Las coordenadas x e y se dividen por 8 ya que hay 8 "pixels" gráficos para cada posición de carácter. Pero hay una desventaja en calcular así la posición de la etiqueta. Como ésta se ha de imprimir entre los extremos de los dos radios, no funcionará bien si el sector es mayor que un semicírculo. Sin embargo, para valores menores, el programa trabaja correctamente y etiqueta los sectores:

GRÁFICO ETIQUETADO



Para cada etiqueta, la línea 240 visualiza el mismo número sobre un fondo coloreado y visualiza el tamaño del sector además de dar los valores tecleados.

Teclee las siguientes respuestas al cuadro de preguntas:

500 - ingresos totales
160 - facturas
100 - seguros
180 - viajes

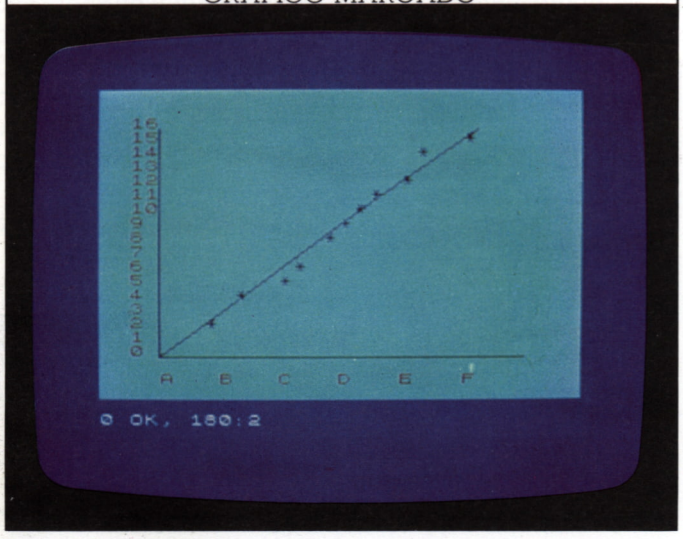
Si prueba estos valores, terminará con un gráfico que tiene un sector negro sin etiqueta. Representa la cantidad de ingresos restantes después de los pagos que ha tecleado.

Puede Vd. adaptar este programa para tener más de tres sectores cambiando el número de entradas (INPUT) y las dimensiones de las matrices. Es posible dibujar con el Spectrum secciones de círculo coloreadas. Sin embargo, si usa rutinas que dibujan con tinta (INK) para colorear sectores en un gráfico, encontrará dificultades en obtener bordes lisos en los sectores. Este problema es debido a que la clave INK, en el Spectrum, funciona mejor para la resolución de textos que para la de gráficos.

GRAFICOS DE PUNTOS Y LINEAS

Como los gráficos son tan buenos para juegos, a menudo sólo se explotan parcialmente para usos más serios. Vd. puede usar su Spectrum para obtener gráficos de alta resolución que muestren cualquier información que se pueda dar en forma numérica. Aunque los gráficos de sectores son útiles para mostrar la división de algo en partes, también muestran la relación de dos series de elementos. Vea la siguiente pantalla:

GRÁFICO MARCADO



Aun sin ser un matemático puede deducir información útil de este gráfico: el valor de la ordenada, y, aumenta uniformemente con el de la abscisa, x.

Cómo construir un gráfico

El programa que crea el gráfico anterior ha de dibujar los ejes de coordenadas, etiquetarlos, marcar los puntos y finalmente dibujar una línea:

PROGRAMA DE GRÁFICO MARCADO

```
  10 BORDER 1: PAPER 5: CLS
  20 DATA "A","B","C","D","E","F
"
  30 DATA 16,7,14,9,13,12,12,13,
10,15,9,16,8,17,7,18,6,20,4,21,3
,24
  40 PLOT 224,24: DRAW -192,0: D
RAW 0,128
  50 FOR c=4 TO 24 STEP 4
  60 READ m$
  70 PRINT INK 2;AT 20,c;m$
  80 NEXT c
  90 LET l=0
 100 FOR r=18 TO 2 STEP -1
 110 PRINT INK 2;AT r,2;l
 120 LET l=l+1
 130 NEXT r
 140 FOR n=1 TO 11
 150 READ r,c
 160 PRINT AT r,c;"*"
 170 NEXT n
 180 PLOT 32,24: DRAW 168,128

0 OK, 0:1
```

La colección de información está contenida en DATA en la línea 30. La línea 40 dibuja los dos ejes del gráfico. Los dos bucles que siguen (entre las líneas 50 y 80, y entre las 100 y 130) visualizan las etiquetas a lo largo de cada eje. El primer bucle toma una a una las letras del DATA de la línea 20 y las pone en el eje horizontal en posiciones determinadas por c en la línea 70. Esta aumenta en pasos de 4 (línea 50) para dejar separación entre cada dos letras. La línea 110 pone números (que por estar en secuencia no necesitan almacenarse en una línea DATA) en el eje vertical, dados por la línea 100.

Cómo programar gráficos "a la medida"

La principal desventaja del programa anterior es que dará siempre el mismo gráfico. Aunque puede variar la información de las líneas DATA, es laborioso modificar la pantalla. Vd. deseará un programa que le permita teclear cualquier coordenada mientras se procesa el programa. Además Vd. deseará no tener que trasladar las coordenadas del gráfico en coordenadas de la pantalla del Spectrum antes de usarlas; el programa debería poder hacer esta conversión.

Esto lo consigue el siguiente programa. Aunque produce ejes que tienen etiquetas fijadas, Vd. puede teclear las coordenadas que desee, siempre que caigan dentro de los límites del gráfico.

La línea 20 dibuja los ejes, como antes; las líneas 30 a 50 ponen las etiquetas al eje x. El programa usa bucles

PROGRAMA DE GRÁFICO VARIABLE

```
  10 BORDER 2: PAPER 6: INK 1: C
LS
  20 PLOT 224,24: DRAW -192,0: D
RAW 0,128
  30 FOR c=4 TO 28 STEP 4
  40 PRINT AT 19,c;c-4
  50 NEXT c
  60 LET n=0
  70 FOR r=18 TO 2 STEP -2
  80 PRINT AT r,2;n
  90 LET n=n+4
 100 NEXT r
 110 PRINT AT 0,2;"Y"
 120 PRINT AT 20,31;"X"
 130 INPUT "X=";h,"Y=";t
 140 BEEP 0.2,20
 150 PRINT AT 18-16*t/32,4+h;"*"
 160 GO TO 130



0 OK, 0:1
```

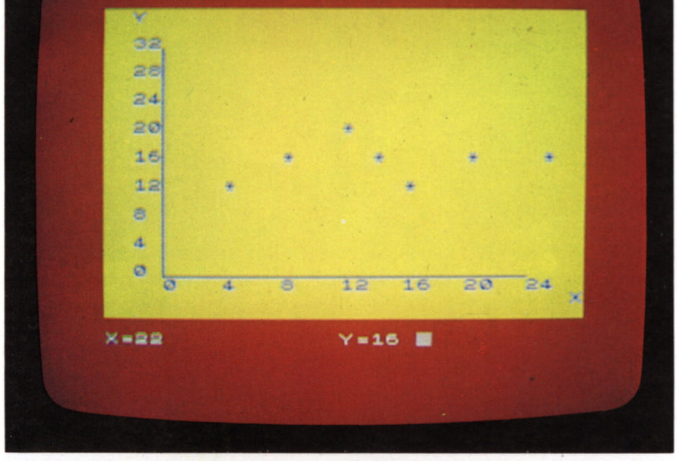
FOR...NEXT para determinar cuáles son las etiquetas y dónde han de ir. Aquellas podrían representar horas. El programa utiliza bucles FOR...NEXT para determinar las etiquetas y su posición. Como las posiciones de las etiquetas horizontales van del 4 al 28 y los valores del

0 al 24, la línea 40 del programa utiliza el número de la columna para poner la etiqueta en posición.

Las líneas 60 a 100 etiquetan el eje vertical. Aquí están elegidas para que representen una gama de temperaturas en centígrados.

La línea 130 le pide introducir un par de coordenadas. Teclee una hora, pulse ENTER y teclee una temperatura, seguida de ENTER. El ordenador suena y la línea 150 marca un asterisco en la posición de la pantalla:

PANTALLA DE GRÁFICO VARIABLE



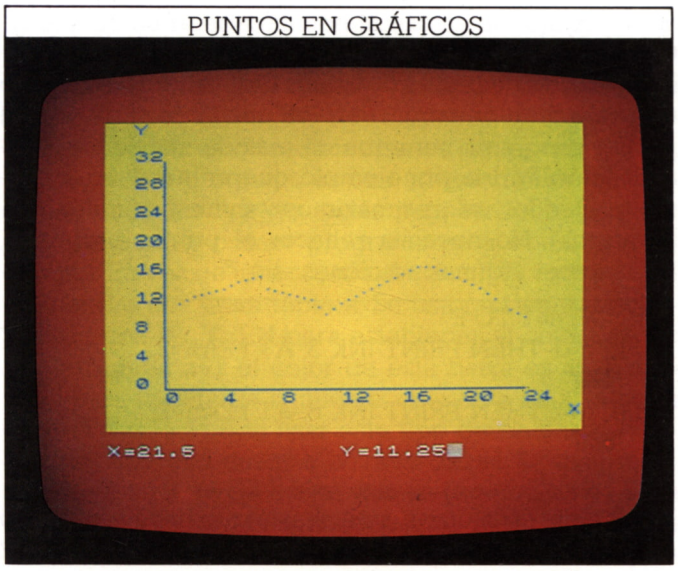
Cómo cambiar la pantalla

El gráfico visualizado es muy basto, ya que sólo pueden marcarse asteriscos en las 384 posiciones (24 col.×16 fil.) dentro del área gráfica. Para un mayor detalle, puede sustituir la línea 150 por la:

```
150 PLOT 32+192*h/24,24+128*t/32
```

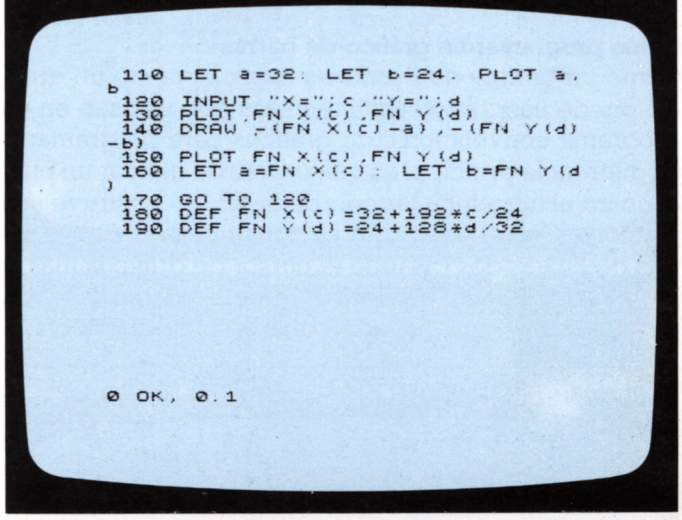
Esta produce una pantalla con puntos, en una red gráfica, en vez de asteriscos, permitiendo mayor resolución:

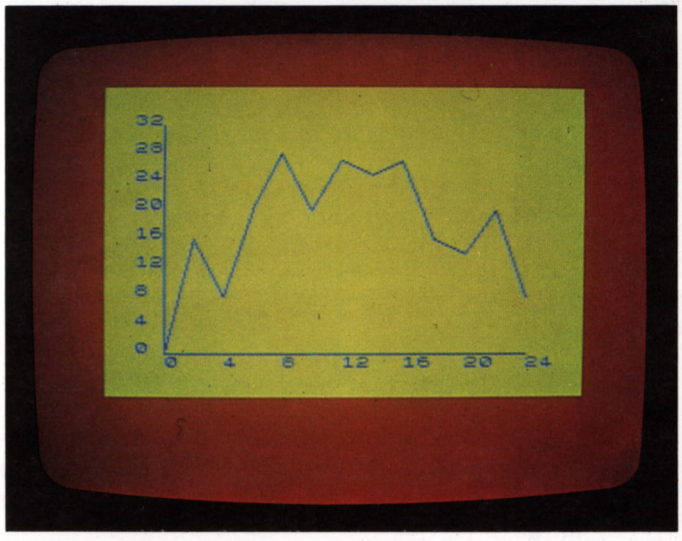
PUNTOS EN GRÁFICOS



Cambiar el programa para que produzca líneas unidas, y no puntos aislados, es un poco más difícil. Lo que Vd. necesita es decirle al ordenador que marque un punto, dibuje desde él al anterior y después marque de nuevo el segundo punto para que, a su vez, pueda volver dibujando. Este tipo de programas es ideal usando funciones. Sin ellas, las líneas del programa tendrían que repetir un par de cálculos pesados para establecer las coordenadas. Estos son los cambios de líneas y adiciones necesarios para producir el gráfico de líneas, junto con una pantalla:

MODIFICACIONES DE GRÁFICOS DE LÍNEAS





Las líneas 180 y 190 definen dos funciones; convierten las coordenadas horizontal y vertical del gráfico en coordenadas de la pantalla de manera que el programa puede marcar (PLOT) y dibujar (DRAW) con ellas. La línea 110 fija el primer punto (a,b) en la esquina inferior izquierda del gráfico. Después, cada vez que Vd. teclea un par de coordenadas, el programa las convierte; marca un punto en x,y; dibuja hacia atrás a a,b; marca x,y de nuevo; y en la línea 160, hace a,b igual a x,y.

GRAFICOS DE BARRAS

La información gráfica puede presentarse de formas diferentes; los gráficos sirven para mostrar tendencias, mientras que los gráficos de barras son útiles para mostrar diferencias en niveles.

El nombre de "barras" procede de que la información, en vez de visualizarse como puntos, se da como columnas cuyas alturas corresponden con el tamaño o nivel de la variable representada. Estos gráficos se usan, por ejemplo en televisión, para mostrar los cambios de moneda, los números de votos en elecciones, etc.

Cómo programar un gráfico de barras

Como un gráfico de barras es esencialmente un gráfico, puede usar las mismas técnicas que se usan en un programa convencional de gráficos para programarlo. La diferencia principal es que en vez de marcar un punto único, al introducir las coordenadas se construye una columna. Con el Spectrum, se construyen con unas series de cuadrados (usando el cursor de gráficos más la tecla 8 y CAPS SHIFT). Después puede usarse INK para dar color. El siguiente programa emplea esta técnica:

PROGRAMA DE GRÁFICO DE BARRAS

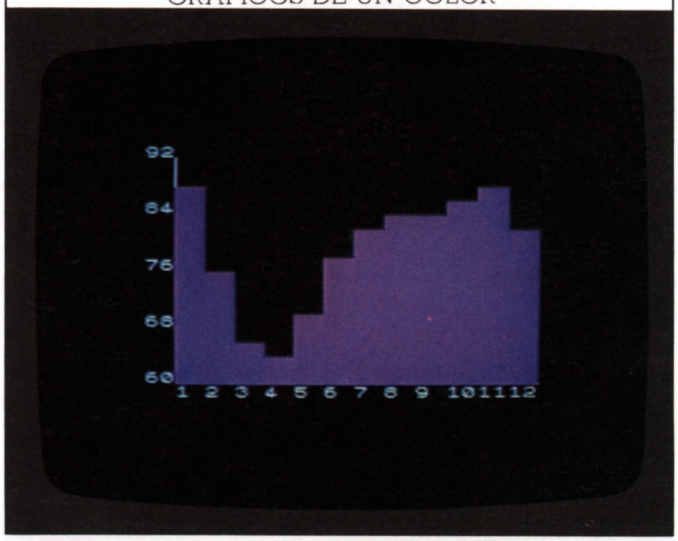
```
  10 BORDER 0: PAPER 0: INK 7: C
LS
  20 PLOT 224,24: DRAW -192,0: D
RAW 0,128
  30 FOR c=4 TO 26 STEP 2
  40 PRINT AT 19,c;(c-2)/2
  50 NEXT c
  60 LET n=60
  70 FOR r=18 TO 2 STEP -4
  80 PRINT AT r,2;n
  90 LET n=n+8
 100 NEXT r
 110 FOR m=1 TO 12
 120 INPUT (m);"=? ";t
 130 BEEP 0.2,10
 140 FOR r=18 TO 18-(t-60)/2 STE
P -1
 150 PRINT INK 3;AT r,2*m+2;"■■"
 160 NEXT r: NEXT m


0 OK, 0:1
```

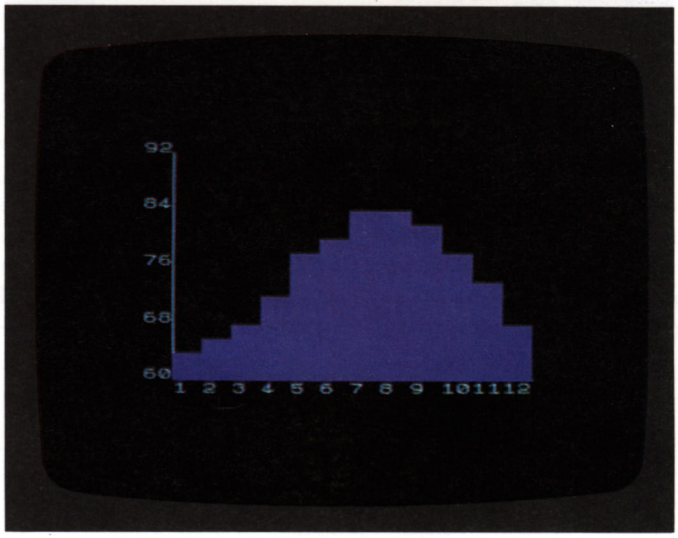
Los ejes se dibujan en la línea 20, en las mismas posiciones que los gráficos de páginas anteriores. El bucle FOR... NEXT en las líneas 30 a 50 etiqueta el eje x con los números 1 a 12, que podrían representar los meses del año. Si las columnas, bajo las cuales han de colocarse las etiquetas, van de 4 a 26 (STEP 2), el número de mes está dado por:

```
mes=(columna-2)/2
```

Pruébelo — la columna 4, al comienzo del eje, equivale al mes 1. La columna 26, al final del eje, equivale al mes 12. Hay 12 escalones (STEP) en total. Aquí tenemos el programa en acción:

GRÁFICOS DE UN COLOR





Combinación de gráficos

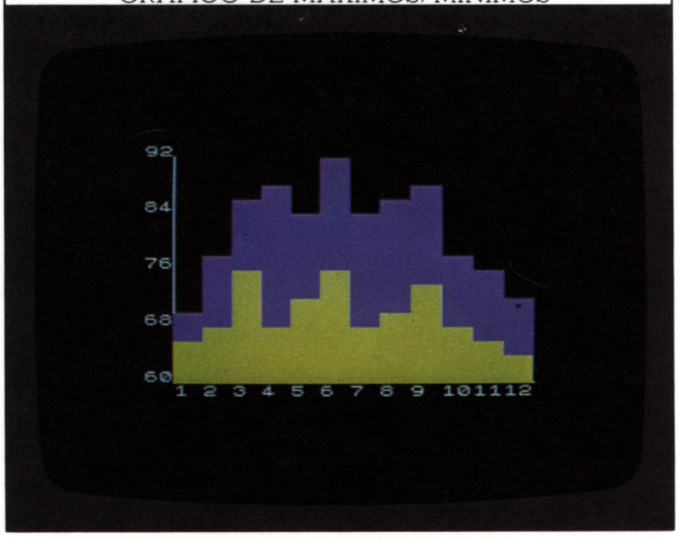
Los gráficos de barras descritos hasta ahora tienen sólo una lista de elementos. Pero es posible reorganizar el primer programa para que dé más de una serie de información. Podría, por ejemplo, querer incluir en el mismo gráfico los valores máximos y mínimos (como temperaturas). No necesita rehacer el primer programa, basta hacer algunas adiciones:

```
105 FOR n=1 TO 2
150 IF n=1 THEN PRINT INK 3; AT r,2*m+2;
    "■■"
155 IF n=2 THEN PRINT INK 6; AT r,2*m+2;
    "■■"
170 NEXT n
```

Esto se procesa como antes hasta que haya terminado de teclear la primera serie de datos. Después hace n=2

y pone las columnas de cuadrados amarillos en vez de los magenta. La segunda serie de datos debe componerse de valores menores que los de la primera; si no, el gráfico amarillo tapará el magenta:

GRÁFICO DE MÁXIMOS/MÍNIMOS



Cuando comience a introducir la segunda serie de elementos, verá que el eje x, y parte del y del gráfico, desaparecen. Puede resolver esto fácilmente redibujando los ejes cada vez que se dibuja una columna, así:

```
156 INK 7:PLOT 224,24:DRAW -192,0:DRAW
  0,128
```

Necesitará visualizar las columnas como PAPER. Esto retarda el programa considerablemente, y puede preferir redibujar los ejes sólo una vez al final, introduciendo la línea anterior como la número 180.

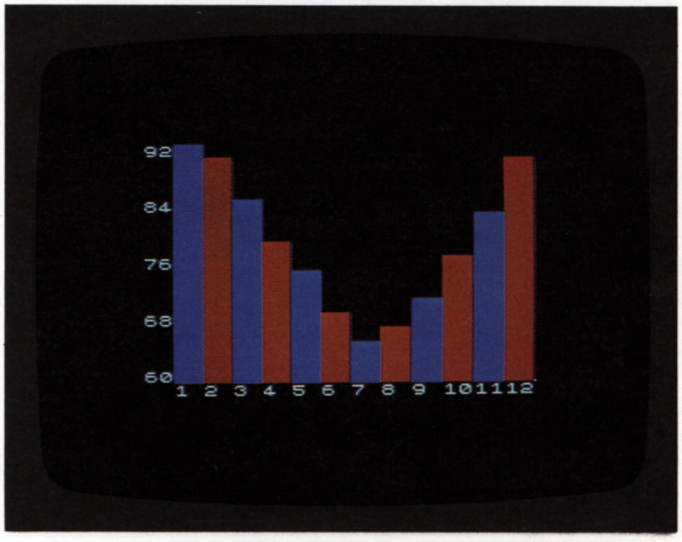
Separación de barras cambiando INK

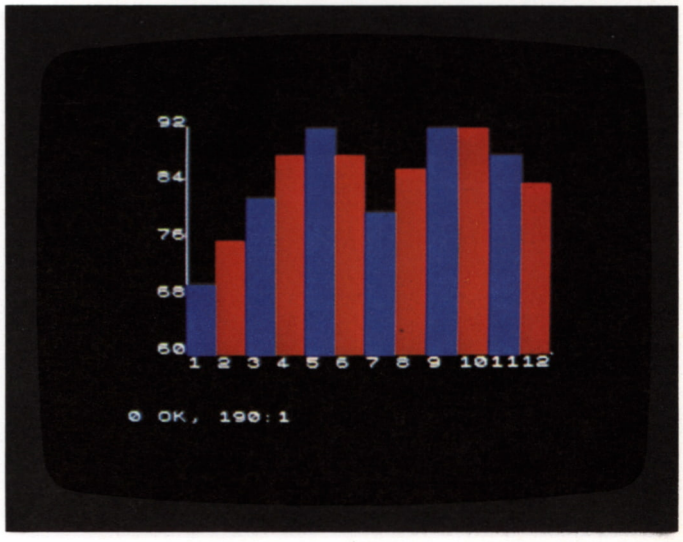
Uno de los problemas de las pantallas visualizadas antes es no poder distinguir cada barra en el gráfico, dificultando su referencia con la escala del eje x. Puede resolverlo usando de nuevo dos colores, pero ahora alternándolos a medida que se visualizan las barras de una serie de entradas (INPUT). Entonces, es bastante fácil ver los valores correspondientes del eje x.

El siguiente programa es una adaptación del primero. Quitando las líneas usadas para la doble pantalla, puede editar el primer programa para obtener el siguiente. En vez de tener fijado el color INK, éste se controla ahora con la variable a. Se usa un bucle en conjunción con IF... THEN para establecer el color (azul o rojo). Cuando a=1 el color de esta barra es azul; después para la entrada siguiente, cuando a es 2, el color cambia a rojo. Puede usar este tipo de bucle de cambio de color con cuantos colores desee. Si quiere aumentar el número de barras que se pueden hacer para que aparezcan en un gráfico, puede reducir el ancho de cada barra usando un único cuadrado. El programa tendría que cambiarse para cambiar las posiciones de visualización. Aquí hay un programa con gráfico de dos colores y algunas pantallas ejemplo que podría producir:

PROGRAMA DE GRÁFICO DE DOS COLORES

```
  10 BORDER 0: PAPER 0: INK 7: C
LS
  20 PLOT 224,24: DRAW -192,0: D
RAW 0,128
  30 FOR c=4 TO 26 STEP 2
  40 PRINT AT 19,c;(c-2)/2
  50 NEXT c
  60 LET n=60
  70 FOR r=18 TO 2 STEP -4
  80 PRINT AT r,2;n
  90 LET n=n+8
 100 NEXT r
 110 LET m=1
 120 FOR a=1 TO 2
 130 INPUT (m);"=? ";t
 140 BEEP 0.2,10
 150 FOR r=18 TO 18-(t-60)/2 STE
P -1
 160 PRINT INK a;AT r,2*m+2;"■"
 170 NEXT r:
 180 LET m=m+1: NEXT a
 190 IF m<12 THEN GO TO 120

0 OK, 0:1
```





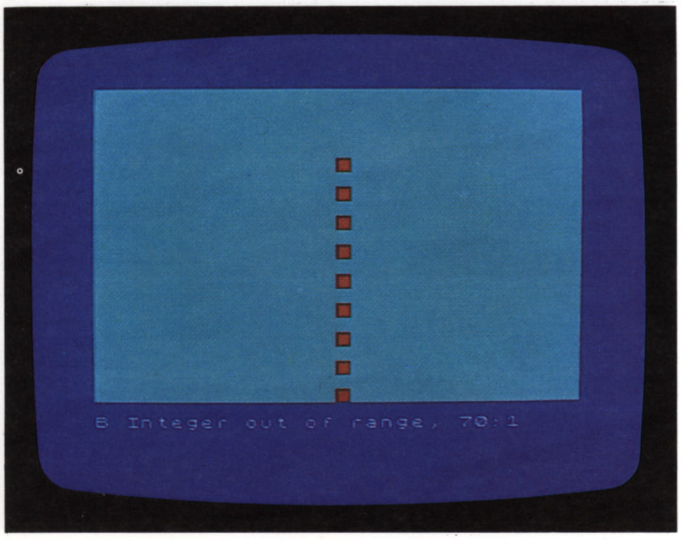
GRAFICOS CON GRAVEDAD

En las páginas 18-19 ya vio cómo se usan SIN y COS para obtener curvas matemáticas (como aparecen a veces en el mundo real), usando los propios mandatos. Pero si quiere que el ordenador simule algo que se mueve de forma realista, le ayudará mucho la comprensión de cómo se mueve en la vida real, cuando lo intente en la pantalla.

Consideremos una simulación sencilla usando un balón que rebota y recorramos las etapas necesarias para construir diferentes tipos de programas. En las páginas 8-9, vio cómo puede usarse IF... THEN para "rebotar" un balón en líneas rectas a velocidad constante. Sin embargo, un balón no se mueve en línea recta. En la siguiente pantalla, un corto programa le muestra cómo podría empezar simulando una caída más real (la pantalla inferior incluye imágenes anteriores normalmente borradas):

PROGRAMA DE CAÍDA SIMPLE

```
 10 BORDER 1: PAPER 5: INK 2: C
LS
 20 LET r=5: LET c=16: LET v=1
 30 PRINT AT r,c;"■"
 40 PAUSE 100
 50 PRINT AT r,c;" "
 60 LET r=r+v
 70 PRINT AT r,c;"■"
 80 PAUSE 5
 90 GO TO 50


0 OK, 0:1
```



Los objetos que caen están influenciados por diversas fuerzas —gravedad, resistencia del aire, fricción superficial y algo llamado "coeficiente de restitución", que los hace moverse de forma compleja. Sin embargo, no hay que ser físico para escribir un programa de "rebote" más realista. Si deja caer un balón, cae al suelo y rebota, eso es todo lo que necesita saber para simular el rebote en la pantalla.

En el programa anterior, la línea 30 visualiza el "balón" cerca de lo alto de la pantalla. Después de 2 segundos de pausa, el "balón" comienza a caer. La línea 50 lo borra. Después, el número de fila se aumenta en 1 y, finalmente, el "balón" se visualiza de nuevo.

Movimiento en dos direcciones

Si procesa este programa, encontrará que aunque el "balón" cae al fondo de la pantalla, su movimiento no parece muy real. También termina con un mensaje de error cuando el "balón" cae fuera del fondo de la pantalla. El siguiente programa mejora la pantalla haciendo que el "balón" también se mueva lateralmente:

PROGRAMA DE CAÍDA LATERAL

```
 10 BORDER 1: PAPER 5: INK 2: C
LS
 20 LET r=5: LET c=16: LET h=1:
LET v=1
 30 PRINT AT r,c;"■"
 40 PAUSE 100
 50 PRINT AT r,c;" "
 60 LET r=r+v: LET c=c+h
 70 PRINT AT r,c;"■"
 80 PAUSE 5
 90 GO TO 50


0 OK, 0:1
```

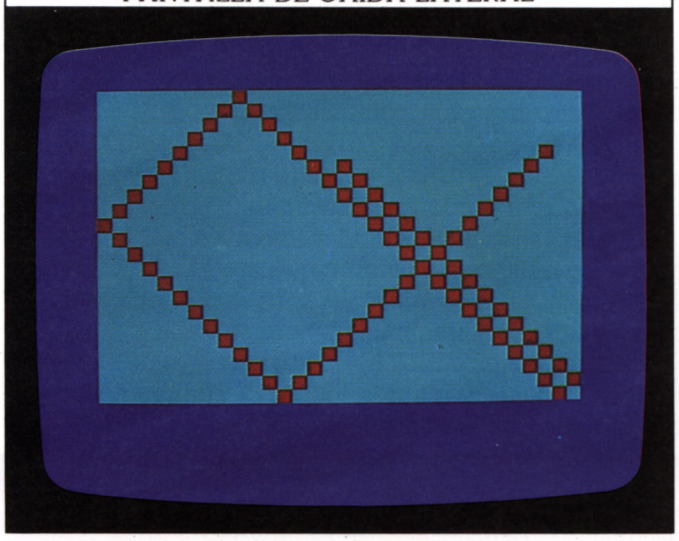
La variable h representa el cambio de posición horizontal y v el de la vertical. En cada bucle, se suma v al número de fila y h al de columna. Ahora es fácil modificar el movimiento en cualquier dirección. Por ejemplo, puede hacer que el "balón" rebote añadiendo:

```
65 IF c=0 OR c=31 THEN LET h=-h: BEEP
0.05,20
66 IF r=0 OR r=21 THEN LET v=-v: BEEP
0.05,20
```

Si suprime las líneas que borran el "balón" cuando se mueve, obtendrá una pantalla como esta. Note que el

"BEEP" suena sólo cuando el "balón" da con el borde de la pantalla:

PANTALLA DE CAÍDA LATERAL



Simulación de la gravedad

Aunque ahora el "balón" rebota alrededor de la pantalla, aún no parece completamente real, ya que no imita los efectos de un objeto que cae por la fuerza de la gravedad.

Puede añadir una "fuerza" como la gravedad, que actúe en cualquier dirección o que cambie de dirección durante el proceso del programa. La gravedad actúa hacia abajo, acelerando la caída del "balón" desde lo alto al fondo de la pantalla. Cuando rebota desde el fondo, debe frenar la subida volviéndolo a hacer caer al fondo. El siguiente programa imita este efecto. Teclee el siguiente listado:

PROGRAMA DE BALÓN QUE REBOTA

```
  10 BORDER 1: PAPER 0: INK 7: C
LS
  20 DATA 48,120,252,252,120,48,
0,0
  30 FOR n=0 TO 7
  40 READ x
  50 POKE USR "a"+n,x
  60 NEXT n
  70 LET r=5: LET c=16: LET h=1:
LET v=1
  80 PRINT AT r,c;"A"
  90 PAUSE 100
 100 REM PRINT AT r,c;" "
 110 LET v=v+0.2
 120 LET r=r+v: LET c=c+h
 130 IF c<1 OR c>30 THEN LET h=-
h: BEEP 0.05,20
 140 IF r<1 OR r>20 THEN LET v=-
v: BEEP 0.05,40
 150 PRINT AT r,c;"A"
 160 PAUSE 5
 170 GO TO 100

0 OK, 0:1
```

En este programa, el balón es un carácter definido por el usuario. En la línea 110 se añade el factor de gravedad; la suma de 0,2 a v significa que el cambio en r —posición vertical— ya no es constante; aumenta en cada bucle, acelerando el balón. Cuando el balón está en el fondo de la pantalla se invierte su dirección (línea 140) y v llega a ser negativo, decreciendo el número de fila, además, el factor aditivo de gravedad en la línea 110 hace v cada vez menos negativo retardando el progreso hacia arriba del balón hasta parar su movimiento vertical; v vuelve a ser positivo y el balón comienza a moverse de nuevo hacia abajo.

Esta pantalla muestra cómo se mueve el balón con el programa:

PANTALLA DE BALÓN QUE REBOTA

El balón rebota como antes, pero ahora no alcanza la misma altura en cada rebote: decrece gradualmente, permaneciendo invariable su movimiento horizontal. El resultado es un ejemplo aproximado de una curva llamada "parábola". El balón rebotará en la línea del fondo de la pantalla, como lo haría un balón en la realidad.

De la misma forma que el movimiento vertical puede ser modificado por una "fuerza", Vd. puede modificar el movimiento horizontal. Da la impresión de un objeto que no sólo cae por la gravedad sino que también es empujado por un fuerte viento.

La curva que el balón hace durante este programa no es continua, debido a que el balón es un carácter de texto, y su movimiento está limitado a las 32×22 posiciones en la pantalla. Si quiere obtener un rebote más limpio, experimente con el mandato PLOT. Se producirá un punto único en unas coordenadas gráficas, dando una curva más nítida sobre la red de gráficos de 256×176. Sin embargo, para hacerlo tendría que modificar el programa de manera que las posiciones de caracteres en todas las líneas se convirtiesen en coordenadas de gráficos. Si Vd. consulta la página 59 verá que no es tan difícil. Como un punto único no es tan fácil de seguir, es más sencillo dejar todas las posiciones marcadas en la pantalla para construir una serie de curvas de gravedad, siguiendo el camino del punto cuando cae al suelo.

PROGRAMAS DE JUEGOS 1

Las seis páginas siguientes le ayudarán a programar juegos, mostrándole cómo reunir todas las fases para hacer un listado completo. La escritura de estos programas requiere alguna planificación cuidadosa antes de empezar a escribir las líneas. Para empezar, necesita decidir qué tipo de juego desea. Muchos juegos combinan la habilidad con un elemento de suerte (dados, naipes, etc.), y muchos tienen una serie de fases diferentes, las cuales le enfrentan con una serie diferente de problemas.

Para planear un juego, lo mejor es empezar dibujando un boceto de la pantalla marcando los colores y las posiciones de cualquier carácter o modelo fijado. Querrá volver a este boceto cuando escriba su programa.

Después, puede dibujar un diagrama que muestre las etapas de programa y el orden en que aparecerán. No es necesario un gráfico detallado —basta una lista de etapas conectadas con flechas para indicar su orden. Un programa completo de juego será más complicado que los programas que haya escrito hasta ahora, por lo que es conveniente diseñar el programa antes de teclearlo. Es más fácil borrar una flecha o un par de líneas a lápiz, en el plan, que redisponer líneas en la pantalla, si al procesar el programa encuentra que no funciona.

Tecleado de la fase 1

Con el juego de esta página, se ha completado la etapa de planificación y ahora puede teclear la primera parte del programa de dos fases. El listado que sigue es para un juego práctico —uno que todos podrían jugar sin conocimiento previo del programa o del ordenador. Debajo está la primera pantalla del programa. Esta fase del juego implica disparo a una nave espacial móvil:

FASE 1 - PANTALLA 1

```
  10 REM Pot shots
  20 BORDER 0: PAPER 5: CLS
  30 DATA 90,153,60,90,126,90,18
9,255
  40 DATA 60,126,90,255,165,126,
165,255
  50 FOR n=0 TO 7
  60 READ a,b
  70 POKE USR "c"+n,a
  80 POKE USR "d"+n,b
  90 NEXT n
 100 DEF FN t()=((65536*PEEK 236
74+256*PEEK 23673+PEEK 23672)/50
)
 110 LET n=0: LET f=0: LET q=0:
LET T=FN t()
 120 LET r=INT (RND*15): LET c=I
NT (RND*28)
 130 LET l=17: LET m=16
 140 CLS : LET n=n+1: IF n=6 THE
N GO TO 400
 150 LET h=0: LET a=1

scroll?
```

El programa le da una base de láser que Vd. puede mover a izquierda o derecha. Sólo puede disparar verticalmente. Unas naves se le aproximan una a una y Vd. debe destruirlas para continuar.

Los DATA para nave y base de láser definidas por el usuario se «leen» por el bucle de las líneas 50 a 90. El tiempo empleado para completar el juego, que se usará más tarde para calcular la puntuación, se define en la línea 100 con la técnica empleada en la página 21. Algunas de las variables usadas en el programa se ponen en sus valores iniciales con la línea 110.

El programa es imposible de descifrar si no sabe qué representan las variables. La tabla siguiente resume qué significa cada una:

FASE 1 - VARIABLES

La primera fase del juego usa un total de catorce variables diferentes para controlar gráficos y registrar impactos.

Variable(s)	Función o significado
r,c	Fija coordenadas de fila y columna de la nave
l,m	Fija coordenadas de fila y columna de base láser
h	Registra los impactos láser
f	Registra el total de impactos láser sobre el blanco
q	Registra el número de disparos del láser
x,y	Punto de partida del rayo láser
a	Fija el cambio de posición de la nave (=1 de izquierda a derecha, en caso contrario =-1)
g	Fija la posición en columna de explosión de una mina
T	Registra el tiempo empleado para terminar el juego cuando se resta de FNt()
n,p	Variables generales

La línea 120 fija el punto inicial aleatorio para la nave. La línea 130 inicia su base láser en la mitad de la fila 17. En la línea 140, n registra el número de ataques de la nave. Cuando ha habido cinco ataques, n será igual a 6; entonces el programa salta a la línea 400 y calcula la puntuación.

La segunda pantalla del programa contiene un número de líneas que toman decisiones y después dirigen el programa a subrutinas posteriores. Verá cuando avance en el listado que los números de línea saltan a veces más de 10. Es debido a que para los números de líneas de subrutinas es más fácil recordarlas como múltiplos de 100.

Cuando teclee las líneas de la segunda pantalla, recuerde que D y C en las líneas 170 y 180 representan caracteres definidos por el usuario. Esto significa que necesitará conmutar al cursor de gráficos para que no se visualicen como letras (después de procesado el programa, pueden listarse en la forma en que aparecen en el juego). La segunda pantalla aparece como esta:

FASE 1 - PANTALLA 2

```
 160 PAUSE 100+RND*100
 170 PRINT INK 1;AT l,m;" D "
 180 PRINT INK 2;AT r,c;" C "
 190 BEEP 0.05,25
 200 IF INKEY$="z" THEN LET m=m-
1
 210 IF INKEY$="x" THEN LET m=m+
1
 220 IF m<0 THEN LET m=0: IF m>3
1 THEN LET m=31
 230 IF INKEY$="m" THEN GO SUB 5
00
 240 IF h=1 THEN GO TO 120
 250 LET c=c+a
 260 IF c=29 OR c=0 THEN LET a=-
a
 270 LET p=INT (RND*20+1)
 280 IF p=3 THEN GO SUB 600
 290 GO TO 170
 400 LET s=FN t()-T)/5+q*2-10*f
 410 STOP

scroll?
```

Después de una pausa aleatoria (línea 160), la base láser y la nave aparecen de nuevo. Las líneas 200 y 210 le permiten mover su base a uno u otro lado y la línea 220 la detiene, desapareciendo del lado de la pantalla. Si pulsa la tecla M, el programa salta a la rutina de "fuego", en la línea 500. Si ésta registra un impacto, el programa retrocede a la línea 120 y comienza de nuevo.

Las líneas 250 y 260 controlan el movimiento de la nave. Las líneas 270 y 280 hacen que el programa salte a la rutina de mina —que está posicionada por una variable listada en la tabla de la página opuesta— en la línea 600 una vez en cada 20 movimientos de la nave. Esto se hace escogiendo un número aleatorio de 1 a 20; precisamente uno de estos números accionará la rutina de mina. La línea 290 continúa el mismo ataque retrocediendo a la línea 170.

Las líneas 400 y 410 calculan la puntuación y terminan esta parte del programa. La puntuación está basada en el tiempo empleado en completar el programa, el número de veces que fue disparado el láser y el número de impactos directos. Realmente la puntuación no se usa aquí, pero aparecerá más tarde cuando Vd. desarrolle el juego. Si quiere comprobar que las líneas de puntuación trabajan, procese el programa y después teclee el mandato PRINT s, sin número de línea. Su puntuación se visualizará en la pantalla.

Sección de la subrutina

Finalmente, aquí está la última parte del primer programa. Contiene un par de subrutinas. Las líneas 500 a 550 dibujan y borran el láser usando DRAW y OVER 1. Si m=c el láser ha impactado su blanco. Las líneas 600 a 690 explosionan una mina en la fila 17. La posición de columna de la mina es aleatoria. Si cae sobre la base de láser, el valor q de su puntuación será afectado.

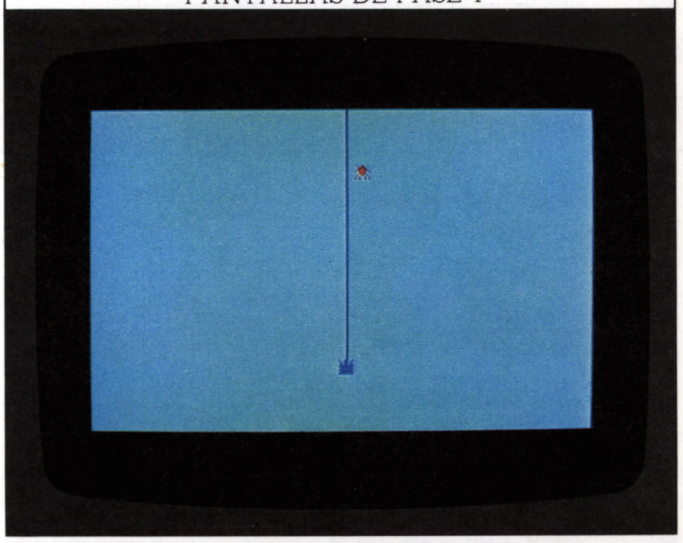
Una vez que Vd. haya tecleado el listado de la pantalla siguiente y procesado esta fase del juego, guárdela en una cinta de manera que esté preparada para combinarla con la parte siguiente del programa:

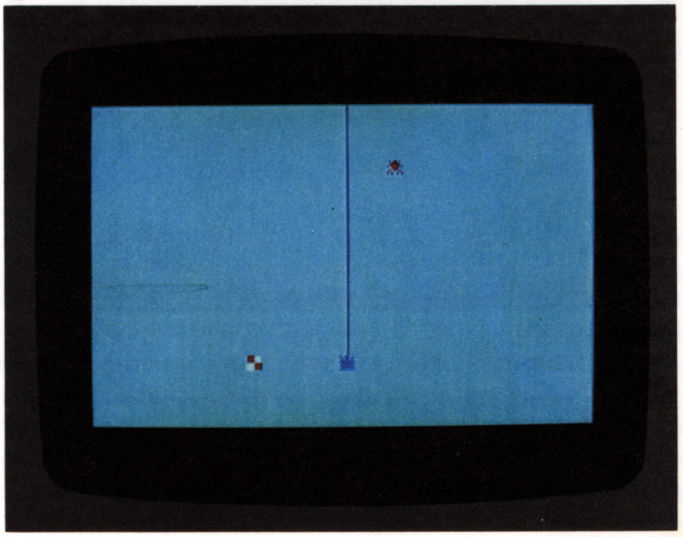
FASE 1 - PANTALLA 3

```
 500 LET x=m*8+12: LET y=40: LET
 q=q+1
 510 PLOT x,y: DRAW INK 0; OVER
1;0,135
 520 BEEP 0.05,50
 530 DRAW INK 0; OVER 1;0,-135
 540 IF m=c THEN BEEP 0.2,0: LET
 h=1: LET f=f+1
 550 RETURN
 600 LET g=INT (1+RND*31)
 610 FOR p=1 TO 10
 620 PRINT INK 2;AT 17,g;"▚"
 630 BEEP 0.02,60
 640 PRINT INK 7;AT 17,g;"▚"
 650 BEEP 0.02,0
 660 NEXT p
 670 IF g=m THEN LET q=q+10
 680 PRINT AT 17,g;" "
 690 RETURN


0 OK, 0:1
```

Aquí tiene el programa en acción. En la primera pantalla, el láser está disparando a la nave, mientras en la segunda ha aparecido una mina:

PANTALLAS DE FASE 1





PROGRAMAS DE JUEGOS 2

En la segunda fase del programa, la escena cambia del aire al mar: un barco lanza una carga de profundidad a un submarino móvil. De nuevo, el objetivo es alcanzar al enemigo. Las instrucciones de puntuación no se usan aún en esta fase, pero están preparadas para entrar en operación cuando se ha tecleado la última fase del juego, uniendo las dos primeras partes.

Como antes, el programa usa un número de variables para controlar movimientos y subrutinas. Hay que explicar estas variables si se quiere entender lo que sucede (en sus programas de juego podría usar líneas REM como recordatorio).

FASE 2 - VARIABLES

La segunda fase utiliza seis variables para controlar los tres objetos animados por el programa.

Variable(s)	Función o significado
c	Después de la línea 1090, fija la posición de columna del barco.
t,d	Coordenadas de fila y columna del submarino.
f	Registra cuándo ha caído la carga de profundidad.
u,e	Coordenadas de fila y columna de la carga de profundidad.

Establecimiento de la escena

La primera sección del programa produce la pantalla coloreada y selecciona algunos números aleatorios:

SECCIÓN PANTALLA/ANIMACIÓN

```
  100 DEF FN t()=((65536*PEEK 235
74+256*PEEK 23673+PEEK 23672)/50
): LET s=0
1010 REM Sub sinker
1020 LET n=0: LET f=0: LET T=FN
t()
1030 BORDER 2: PAPER 4: INK 1: C
LS
1040 IF n=5 THEN GO TO 1400
1050 FOR r=0 TO 4
1060 FOR c=0 TO 31
1070 PRINT AT r,c;"■"
1080 NEXT c
1090 NEXT r
1100 LET c=14
1110 LET t=10+INT (RND*11)
1120 LET d=INT (RND*11)
1130 IF INKEY$="z" THEN LET c=c-
1
1140 IF INKEY$="x" THEN LET c=c+
1
1150 IF c>25 THEN LET c=25

scroll?
```

De momento, ignore la línea 100; en la página siguiente se enterará por qué se ha incluido. Las líneas 1050 a 1090 visualizan un cielo azul sobre un mar verde, trazando filas de cuadrados azules sobre la parte superior del fondo verde del papel. La línea 1100 fija la posición de columna del barco y las 1130 a 1160 controlan su movimiento a través de la pantalla.

El objetivo del juego es impactar el submarino. La tecla M controla el lanzamiento de las cargas de profundidad del barco. El barco también es maniobrable. Si Vd. pulsa la tecla Z, el barco se moverá a la izquierda, y pulsando la X, a la derecha. Todas estas funciones son controladas por INKEY$ para una rápida respuesta. El barco siempre empieza en la mitad de la pantalla. La posición del enemigo es menos predecible. Las líneas 1110 y 1120 fijan el punto de azar inicial del submarino; puede aparecer a casi cualquier profundidad y en cualquier punto a través de la pantalla.

Programa principal y subrutinas

La segunda parte del programa contiene algo del programa principal, junto con subrutinas que son llamadas por el programa principal. Aquellas controlan el movimiento sobre la pantalla y detectan si sus cargas de profundidad han alcanzado el blanco o no.

SECCIÓN DE SUBRUTINAS

```
1160 IF c<0 THEN LET c=0
1170 PRINT PAPER 1; INK 7;AT 3,c
;"▄ "
1180 PRINT PAPER 1; INK 7;AT 4,c
;"▄▄▄▄ "
1190 IF f=1 THEN GO SUB 1310
1200 IF INKEY$="m" AND f=0 THEN
LET f=1: LET s=s+100: GO SUB 130
0
1210 LET d=d+1
1220 IF d>24 THEN PRINT AT t-1,d
+1;" ";AT t,d;" ": LET d=0
1230 PRINT AT t-1,d;"▄▄▄ "
1240 PRINT AT t,d;" ▄▄▄ "
1250 GO TO 1130
1300 LET u=5: LET e=c+2
1310 PRINT AT u,e;" "
1320 LET u=u+1
1330 IF u=21 THEN LET f=0: RETUR
N
1340 PRINT AT u,e;"▄"
```

```
1350 IF u=t AND e>d AND e<d+6 TH
EN BEEP 0.2,0: LET n=n+1: LET f=
0: GO TO 1030
1360 RETURN
1400 LET s=s+FN t()-T
0 OK, 0:1
```

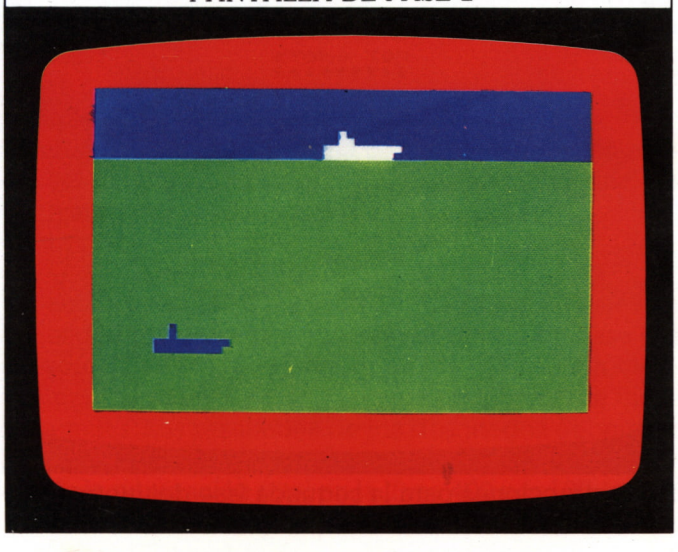
La línea 1200 hace saltar al programa a la rutina de carga de profundidad en la línea 1300 si se pulsa M. La puntuación, s, también se ajusta cada vez que se lanza una carga y está relacionada con el tiempo que ha pasado usando FN t() en la última línea del programa. Las líneas 1210 a 1240 controlan el movimiento y aparición del submarino. La línea 1250 continúa el programa volviéndolo a la línea 1130 para comprobar el teclado por pulsaciones de tecla.

Las líneas 1300 a 1360 hacen que una carga baje por la pantalla. La carga se compone del carácter de gráficos de la tecla 3. Si la carga alcanza el fondo de la pantalla, la línea 1330 vuelve f a su valor original (cero) y retorna al programa principal. Sin embargo, si la posición de la carga coincide con cualquiera ocupada por el submarino (línea 1350), ese ataque está terminado y se empieza uno nuevo. Notará que cuando procesa el programa sólo puede lanzarse una carga cada vez. Si f se ha fijado en 1 por la línea 1200, cuando se lanza una carga, la línea 1190 le impide lanzar otra carga hasta que f sea cero de nuevo. Esto será cierto si la carga alcanza el fondo de la pantalla (línea 1330) o si impacta al submarino (línea 1350).

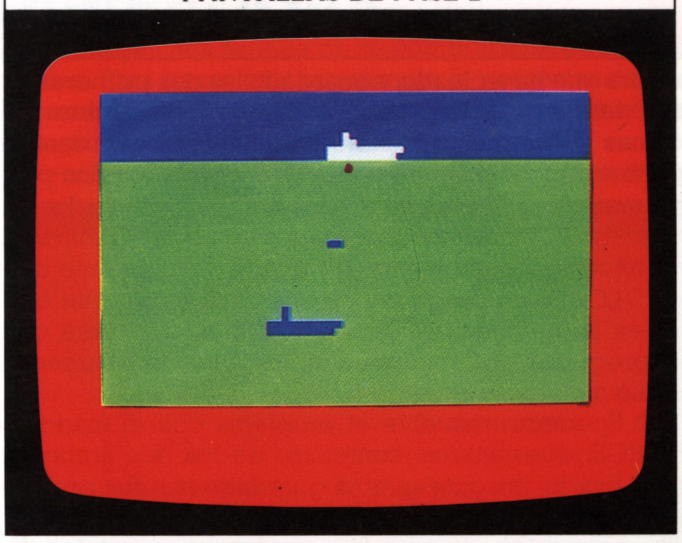
El programa usa gráficos del teclado para construir todos los objetos en esta pantalla y, como verá en las pantallas siguientes, los resultados muestran los bocetos de estos caracteres. Sin embargo, es fácil mejorar el juego usando caracteres definidos por el usuario. Para hacer un barco o submarino, visualice una fila de caracteres juntos. La mayor resolución que pueda entonces alcanzar mejorará considerablemente la pantalla, aunque se necesitarán más líneas de programación.

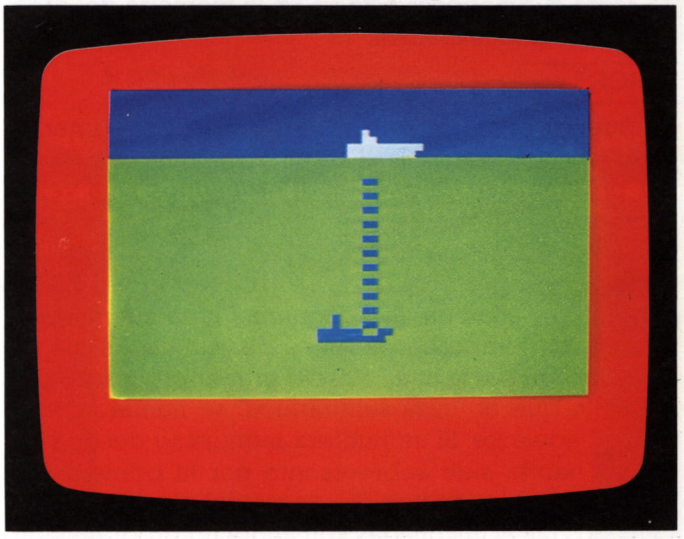
Aquí se muestran algunas pantallas del juego en acción. En la tercera, se ve el camino de una simple carga poniendo un mandato REM en la línea 1310, incapacitando el mandato PRINT que normalmente borra la carga cada vez que ésta se mueve una posición hacia abajo:

PANTALLA DE FASE 2



PANTALLAS DE FASE 2





Podría haberse dado cuenta de que no parece existir aquí ningún medio de borrar las antiguas imágenes no deseadas de barco y submarino antes de visualizar las nuevas imágenes. Como ambos sólo se mueven una posición a uno u otro lado de su posición en curso, pueden borrarse incluyendo un cuadrado blanco (espacio) a cada lado de los gráficos.

La rutina de puntuación

Se usa el tiempo una vez más para calcular la puntuación al final del juego. En la versión final de este programa, se definirá la función tiempo en la primera fase del juego y no es necesaria una segunda definición en esta parte. Sin embargo, si quiere comprobar que el programa funciona adecuadamente, necesita tener la función tiempo de forma que pueda visualizar s. La línea 100 se pone para poder probar la rutina de puntuación. La función tiempo se quitará de nuevo cuando se escriba la versión final del juego.

Cuando haya comprobado que el programa funciona, guárdelo preparado para combinarse con la fase 1.

PROGRAMAS DE JUEGOS 3

Ahora que ha tecleado y guardado las dos primeras fases del juego, está preparado para añadir las instrucciones y completar la parte de programa que dará la puntuación. Lo primero a resolver es que los dos programas se guardaron como dos archivos separados en su magnetófono. Para procesarlos como programa único necesitan combinarse. No puede simplemente cargar (LOAD) un programa de una cinta a otro que está ya en la memoria del ordenador. Si lo hace, verá que el programa en memoria simplemente desaparecerá, como si hubiese pulsado NEW.

El Spectrum resuelve el problema con el mandato MERGE, que junta el contenido de los dos archivos. Para usarlo, imagine que ha guardado los dos programas de las páginas 38-41 en cinta pero que el programa «caza-submarinos» está ya en la memoria del ordenador. Si entonces teclea:

```
MERGE "     "
```

—poniendo entre las comillas el nombre del primer archivo— y después pulsa ENTER y pasa la cinta, el primer programa se cargará en el ordenador. Esta vez el programa «caza-submarinos» no se borrará como sucede con LOAD. Sin embargo, MERGE sólo combina programas si sus números de línea no se solapan. La línea extra (100), añadida al programa "caza-submarinos" para poder comprobar la rutina de puntuación, se sobre-escribirá por la línea 100 del primer programa. Esto es por lo que el "caza-submarinos" se numera desde 1010 en adelante. Si se hubiera numerado desde la línea 10, habría sido sobre-escrito por la primera fase del juego en donde hubiera líneas del mismo número.

Las dos fases fusionadas son ahora un programa único. Como medida de precaución debería ahora guardar el programa conjunto en una cinta. Merece la pena si se ha de desarrollar un largo programa, ya que un borrado accidental podría traducirse en mucho tiempo para rehacerlo.

Adición de las instrucciones del juego

Si procesa un programa fusionado, verá que, aunque es teóricamente un sólo programa, se comporta aún como dos unidades separadas. Al escribir programas en fases, como éste, necesitará hacer una pequeña adaptación al programa final fusionado para hacerlo procesable.

La unión de las dos fases se hace fácilmente. Cambie la línea 410 por:

```
410 GOTO 1000
```

Eso no es un error, aun cuando el programa "caza-submarinos" empieza en la línea 1010. Es para permitirle algún espacio para añadir instrucciones de juego empezando en la línea 1000.

Ahora puede retroceder al principio e iniciar el programa con un cuadro de título que contenga todas las instrucciones que el jugador necesita. Se necesita listar las teclas que controlan el movimiento de objetos en la pantalla, y también decirle al jugador cómo se ha de iniciar el juego, teniendo en cuenta que cuando aparece el mensaje, el programa que contiene el juego ya está en proceso. Aquí hay cuatro líneas que dan las instrucciones de apertura para la fase 1:

LÍNEAS DE INSTRUCCIONES DE FASE 1

```
  1 PRINT AT 2,11;"POT SHOTS";A
T 3,11;"*********";AT 8,5;"Five
alien spacecraft";AT 9,2;"are la
ying mines in your zone";AT 11,2
;"You must destroy the aliens"
  2 PRINT AT 13,2;"************
**************";AT 18,6;"LASER
BASE CONTROLS";AT 19,4;"z:left
x:right  m:fire";AT 21,5;"Press
any key to start"
  3 IF INKEY$<>"" THEN GO TO 3
  4 IF INKEY$="" THEN GO TO 4
```

Las líneas 1 y 2 visualizan el título del juego y explican sus controles. La línea 3 impide al ordenador aceptar sus tecleados RUN y ENTER como activadores para iniciar el juego. La vez siguiente que Vd. pulse una tecla, se rompe la condición de repetir la línea 4, y comienza la primera fase del juego:

INTRUCCIONES DE FASE 1

```
POT SHOTS
*********

Five alien spacecraft
are laying mines in your zone
You must destroy the aliens
**************************

LASER BASE CONTROLS
z:left  x:right  m:fire
Press any key to start
```

Las instrucciones para la segunda fase del juego se insertan de forma similar:

LÍNEAS DE INSTRUCCIONES DE FASE 2

```
1000 PRINT AT 2,11;"SUB SINKER";
AT 3,11;"**********";AT 8,4;"Now
 five submarines have";AT 9,0;"i
nvaded your territorial waters";
AT 11,3;"You can depth charge th
em"
1001 PRINT AT 12,3;"*************
*************";AT 18,5;"SURFACE
SHIP CONTROLS";AT 19,0;"z:left
x:right  m:drop charge";AT 21,5;
"Press any key to start"
1002 IF INKEY$<>"" THEN GO TO 10
02
1003 IF INKEY$="" THEN GO TO 100
3
```

```
0 OK, 0:1
```

```
SUB SINKER
**********

Now five submarines have
invaded your territorial waters

You can depth charge them
*************************

SURFACE SHIP CONTROLS
z:left  x:right  m:drop charge

Press any key to start
```

Puede, por supuesto, usar cualquier tecla que quiera para especificar movimiento en tanto la cambie en el programa. Ahora ninguna fase del juego comienza hasta que el jugador está preparado y pulsa una tecla para empezar.

Completado de la rutina de puntuación

También necesita añadir una rutina de puntuación al final del programa fusionado. Se retiene la línea final de puntuación del "caza-submarinos":

```
1400 LET s=s+FNt()-T
```

Sin embargo, si ha jugado los dos juegos y después ha tecleado PRINT s, habrá notado que la primera fase del juego da un resultado que va desde -20 o así a cientos, pero la segunda fase produce resultados de cientos a miles. Los resultados de los dos juegos deben ser del mismo orden de magnitud. Se consigue multiplicando la puntuación total en la línea 400 por 100, lo que hace la puntuación compatible con la de la fase 2:

```
400 LET s=100*(FNt()-T)/5+q  2-(10*f)
```

¡Esta línea es una prueba de su comprensión de las variables! Para hacer más interesante la presentación de la puntuación, puede añadir unas líneas para convertir esta puntuación numérica en categorías:

RUTINA DE PUNTUACIÓN

```
1410 BORDER 0. PAPER 7: INK 0. C
LS
1420 PRINT AT 6,2;"You have earn
ed the rank of"
1430 IF s<1000 THEN LET a$="COMM
ANDER"
1440 IF s>=1000 AND s<2000 THEN
LET a$="CAPTAIN"
1450 IF s>=2000 AND s<4000 THEN
LET a$="PILOT"
1460 IF s>=4000 AND s<6000 THEN
LET a$="CADET"
1470 IF s>=6000 THEN LET a$="ROO
KIE"
1480 PRINT AT 10,12;a$
```

```
0 OK, 0:1
```

```
You have earned the rank of

PILOT

0 OK, 1480:1
```

Las líneas 1410 a 1480 dividen las puntuaciones en bandas, a cada una de las cuales se le asigna una categoría. Una serie de líneas IF... THEN decide dónde cae su puntuación en la clasificación. Vd. puede cambiar los límites de cada banda para dificultar o facilitar los juegos.

Ahora tiene un juego completo de dos fases con instrucciones, acción y rutina de puntuación. Aunque las dos fases usadas en estas páginas son simples, la manera de combinarlas puede usarse para elaborar juegos propios mucho más complejos. Puede usar MERGE para juntar un número de sub-programas, escrito y probado cada uno por su parte. La única restricción es el tamaño de memoria del ordenador, pero a menos que combine programas muy largos no será un problema.

COMO MEJORAR EL SONIDO

El mandato usado en el Spectrum para producir sonido, BEEP, es muy sencillo. La línea:

```
100 BEEP d,p
```

producirá un sonido en un tono p durante d segundos. Uno de los problemas de usarlo en programas es que para todo mientras suena. Así, si quiere producir largos sonidos en conjunción con movimiento en la pantalla no puede usar un simple mandato BEEP, o el programa se "congelará" mientras dura BEEP. Para enlazar sonido y movimiento en lo posible, se necesita usar BEEPs breves con frecuencia.

Cómo mejorar sonido con animación

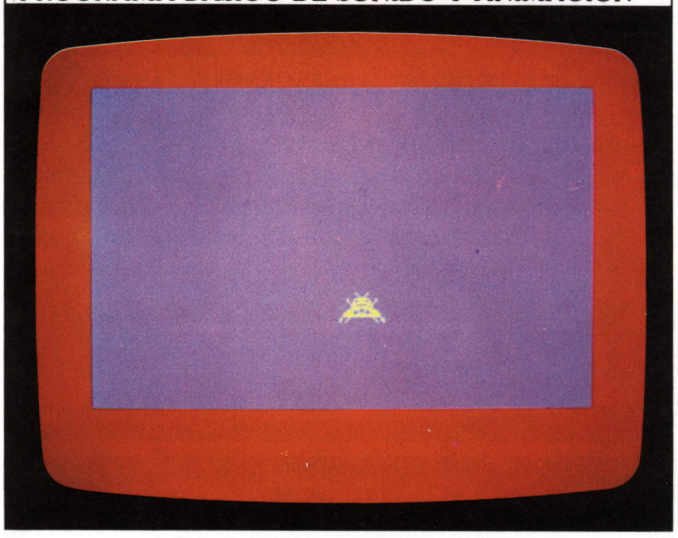
Vd. puede oír BEEP funcionando mal con movimiento si procesa el programa:

PROGRAMA BÁSICO DE SONIDO Y ANIMACIÓN

```
  10 DATA 0,130,0,3,239,128,0,68
,0,15,69,224
  20 DATA 0,40,0,30,40,240,8,254
;32,63,125,248
  30 DATA 9,57,32,63,255,248,5,2
55,64,68,0,68
  40 DATA 3,131,128,200,0,38,2,2
54,128,144,0,18
  50 FOR n=0 TO 7
  60 READ u,v,w,x,y,z
  70 POKE USR "a"+n,u
  80 POKE USR "b"+n,v
  90 POKE USR "c"+n,w
 100 POKE USR "d"+n,x
 110 POKE USR "e"+n,y
 120 POKE USR "f"+n,z
 130 NEXT n
 140 LET r=10: LET c=14
 150 BORDER 2: PAPER 3: INK 6: C
LS
 160 PRINT AT r,c;"      "
 170 PRINT AT r+1,c;" ABC "

scroll?
```

```
 180 PRINT AT r+2,c;" DEF "
 190 PRINT AT r+3,c;"     "
 200 LET a=INT (RND*2)
 210 IF a=0 THEN LET a=-1
 220 LET b=INT (RND*2)
 230 IF b=0 THEN LET r=r+a
 240 IF b=1 THEN LET c=c+a
 250 IF c<0 OR c>27 OR r<0 OR r>
18 THEN GO TO 150
 260 BEEP 0.2,25
 270 GO TO 160



0 OK, 0:1
```

PROGRAMA BÁSICO DE SONIDO Y ANIMACIÓN



El platillo volante está formado por seis caracteres definidos por el usuario dispuestos en dos filas —a, b, c, arriba y d, e, f, debajo, con un espacio a cada lado. Además, se visualiza una fila de espacios encima y debajo del platillo, para que, siempre que se mueva, los espacios borren cualquier imagen previa. La línea 200 da 0 o 1 al azar. La 210 los convierte en -1 o +1, utilizados para cambiar la posición del platillo. La línea 220 da otro 0 o 1 al azar. Si se tiene 0, la fila del platillo se cambia por la variable a; si se tiene 1, se cambia su columna por a. Esto mueve el platillo en la pantalla de forma impredecible. La línea 250 comprueba si el platillo ha alcanzado un borde de la pantalla; caso afirmativo, lo devuelve al centro de la pantalla. El sonido viene dado por la línea 260 en un simple mandato BEEP de una duración de un quinto de segundo. Puede mejorarse añadiendo estas líneas:

NUEVAS LÍNEAS PARA DIVIDIR EL SONIDO

```
195 BEEP 0.005,25
205 BEEP 0.005,30
215 BEEP 0.005,35
225 BEEP 0.005,40
235 BEEP 0.005,45
245 BEEP 0.005,40
260 BEEP 0.005,35



0 OK, 0:1
```

La primera versión es un ejemplo muy pobre de animación. La animación efectiva depende de cambiar desde una pantalla de caracteres a la siguiente tan rápido como sea posible, pero en el programa el BEEP interrumpe el movimiento demasiado tiempo y el platillo se mueve de forma lenta e interrumpida, por lo que realmente el efecto del sonido añade poco.

Añadiendo las líneas de la anterior pantalla puede dividir el BEEP en un número de sonidos más cortos, mejorando la calidad de sonido en el programa. Cada BEEP dura ahora sólo cinco centésimas de segundo. La duración total de todos los BEEPs (0,035 segundos) es menos de un cuarto del BEEP usado anteriormente; así el programa marcha más o menos cuatro veces más rápido que en la primera versión. Además, el sonido dividido en un número de exposiciones separadas permite variar el tono, lo que produce un efecto sonoro mucho más interesante.

Podría también escribir más BEEPs entre las líneas de introducción (10 a 140) para anunciar la llegada del platillo antes de aparecer, pero note que cualquier BEEP escrito entre las líneas 50 a 130 sonará ocho veces debido al bucle n que las conecta.

Cómo enlazar BEEP a las posiciones de la pantalla

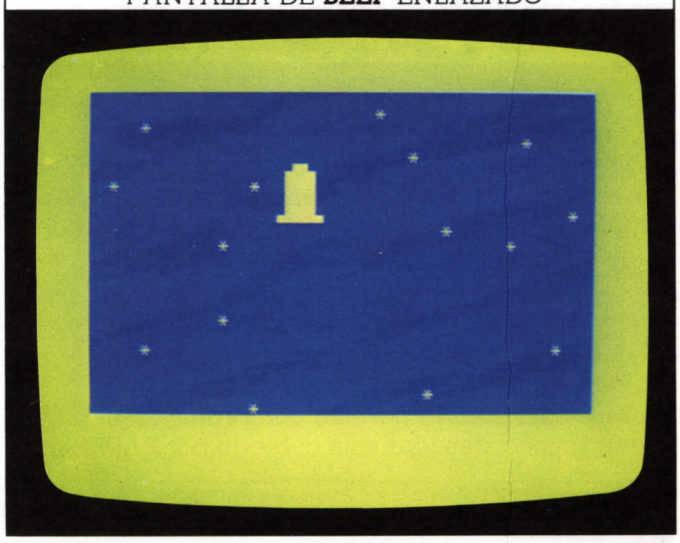
Un medio muy eficaz de producir sonido con animación es enlazar el tono del BEEP a una variable que controla una posición de carácter en la pantalla. Vd. ya se encontró con esto en el programa de "rebote" (páginas 8-9). Enlazar así BEEP es realmente fácil y ahorra muchas líneas de programación. Sólo debe asegurarse de que la variable que controla el BEEP caiga dentro del margen adecuado, y se use para cambiar la duración del tono en el sentido correcto. Puede experimentar estas técnicas en el programa siguiente. Tecléelo y procéselo.

Este programa visualiza un campo de estrellas y después mueve, hacia arriba, un cohete de gráficos del teclado. Las líneas 10 a 60 visualizan el campo de estrellas. Las líneas 70 a 130 visualizan y mueven el cohete. Cuando procesa el programa debe aparecer la pantalla siguiente:

ENLACE DE **BEEP** CON POSICIÓN

```
 10 BORDER 6: PAPER 1: INK 6: C
LS
 20 DATA 1,18,2,3,3,27,4,20,6,1
,6,10,8,30,9,22,10,8,10,26,15,6,
17,3,17,29,20,21,21,10
 30 FOR n=1 TO 15
 40 READ r,c
 50 PRINT AT r,c;"*"
 60 NEXT n
 70 FOR r=21 TO 4 STEP -1
 80 PRINT AT r-4,12;"  "
 90 PRINT AT r-3,12;"  "
100 PRINT AT r-2,12;"  "
110 PRINT AT r-1,12;"  "
120 PRINT AT r,12;"  "
130 NEXT r


0 OK, 0:1
```

PANTALLA DE **BEEP** ENLAZADO



Ahora puede añadir los efectos de sonido al programa anterior. Esta vez su objetivo es producir un sonido extraño de cambio de tono. Sólo se necesita una sentencia BEEP que contenga una variable a, cuyo valor cambie de tono cada vez que se produce el BEEP. Lo inmediato es relacionar el tono al número cambiante de fila. El problema es que el sonido debe aumentar de tono a medida que sube el cohete, pero con ello el número de fila decrece. Así, si se hiciera el tono múltiplo o submúltiplo del número de fila, decrecería al subir el cohete.

Hay una manera de resolver este problema. Pruebe tecleando la línea:

```
125 BEEP 0.05,80-(10*r)/2
```

Ahora, cuanto menor es r, mayor el número de tono. Si la escala de tonos es demasiado grande, multiplicando r por un número menor se reducirá:

```
125 BEEP 0.05,80-(5*r)/2
```

Si los tonos son en conjunto demasiado altos, reduciendo el valor inicial (80) se reducirán los de todos.

```
125 BEEP 0.05,60-(5*r)/2
```

Para hacer más interesante el sonido, puede dividirlo en dos componentes:

```
 75 BEEP 0.025,60-(5*r)/2
125 BEEP 0.025,63-(5*r)/2
```

EL INDICADOR DE LA PANTALLA SPECTRUM

En muchos programas de juegos Spectrum, las áreas de color INK en la pantalla representan objetos que tiene que evitar. Si Vd. choca con ellas, los programas responden con un tipo de penalidad. Pero, ¿cómo programar el ordenador para que decida si un carácter que Vd. mueve en la pantalla ha tocado algo? Podría usar IF... THEN para comprobar coordenadas, pero hay una forma más rápida y sencilla.

El mandato POINT, del Spectrum permite que el ordenador examine cualquier punto en la pantalla y compruebe si es un color PAPER o INK. En la línea:

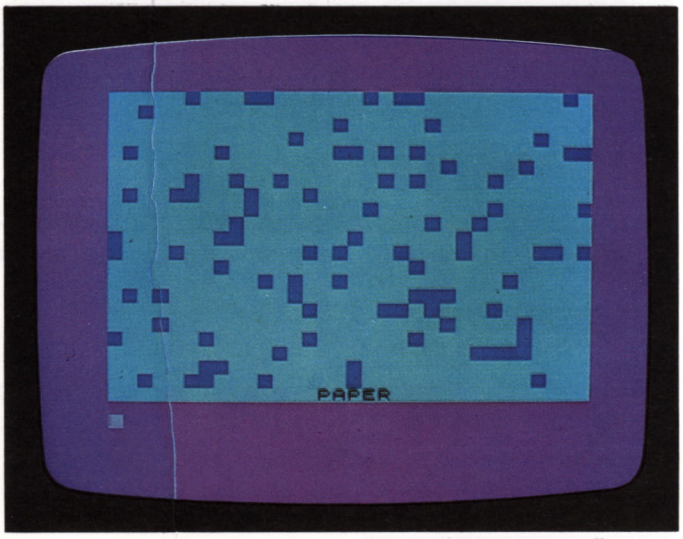
```
100 a=POINT(x,y)
```

a será igual a cero si x,y es un PAPER, o será igual a 1 si x,y es INK.

El siguiente programa produce cuadrados de color al azar, y después le permite comprobar si un particular par de coordenadas está cubierto por PAPER o INK:

COMPROBADOR **PAPER/INK**

```
  10 BORDER 3: PAPER 5: INK 1: C
LS
  20 FOR n=1 TO 100
  30 LET r=INT (RND*21)
  40 LET c=INT (RND*32)
  50 PRINT AT r,c;"■"
  60 NEXT n
  70 INK 0
  80 INPUT x,y
  90 LET a=POINT (x,y)
 100 PLOT x,y
 110 IF a=0 THEN PRINT AT 21,14;
"PAPER"
 120 IF a=1 THEN PRINT AT 21,14;
" INK "
 130 GO TO 80



0 OK, 0:1
```



PANTALLA COMPROBADORA **PAPER/INK**

Las líneas 20 a 60 visualizan cuadrados azul oscuro al azar sobre un fondo azul claro. La línea del fondo de la pantalla se deja en blanco para uso posterior. La línea 70 conmuta a INK negro. La línea 80 espera a que Vd. teclee las coordenadas x,y de cualquier punto.

Teclee una abscisa, x, (0–255), pulse ENTER, y después teclee una ordenada, y, (0–176) y pulse ENTER de nuevo. La línea 90 comprobará si este punto es PAPER o INK. La línea 100 marca un punto negro en x,y para señalarlo. Vea que este programa no funcionaría si las líneas 90 y 100 estuvieran invertidas, debido a que x,y sería siempre INK, que viene justo después de marcar x,y.

El resultado de la línea 90 está visualizado en la fila 21 por las líneas 110 y 120. Si x,y es PAPER, el punto introducido aparece como un pequeño punto negro en la pantalla. Si x,y es INK, todo el cuadrado se vuelve negro debido a que si se cambia el color INK de un simple punto en cualquier posición del carácter, toda la posición del carácter cambia al nuevo color INK.

Cómo usar POINT con movimiento

Ahora que sabe usar POINT, haga una aplicación práctica de este mandato. Probablemente habrá atravesado situaciones en que tiene varios caracteres en la pantalla, pero no quiere molestarse en almacenar sus posiciones y recalcular cuando cualquiera de los caracteres se mueve, con objeto de conocer su posición durante la animación.

El siguiente programa resuelve ese problema. Usa POINT para probar la posición de obstáculos dispersos en la pantalla cuando un carácter rebota entre los bordes de la pantalla. Hay sólo unos pocos obstáculos en esta pantalla, pero puede fácilmente aumentar el número cambiando el margen del bucle entre las líneas 20 y 50:

PROGRAMA DE DRAGAMINAS

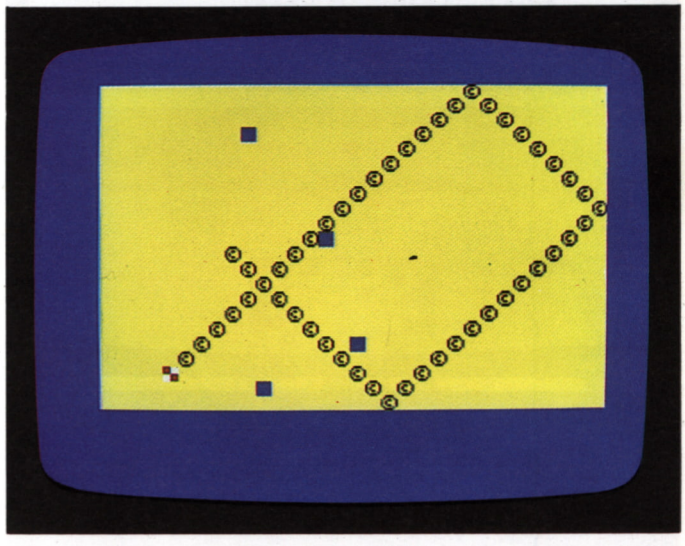
```
  10 BORDER 1: PAPER 6: INK 1: C
LS
  20 FOR n=1 TO 5
  30 LET s=INT (RND*22): LET d=I
NT (RND*32)
  40 PRINT AT s,d;"■"
  50 NEXT n
  60 LET r=10: LET c=7: LET v=1:
 LET h=1
  70 IF r=s AND c=d THEN GO TO 2
0
  80 PRINT AT r,c;" "
  90 IF r=0 OR r=21 THEN LET v=-
v: BEEP 0.1,5
 100 IF c=0 OR c=31 THEN LET h=-
h: BEEP 0.1,29
 110 LET r=r+v: LET c=c+h
 120 LET x=c*8+4: LET y=176-(r*8
+4)
 130 IF POINT (x,y)=1 THEN GO TO
 190

```

```
 140 PRINT INK 0;AT r,c;"©": PAU
SE 3
 150 GO TO 80
 160 FOR n=1 TO 10
 170 FOR a=1 TO 6
 180 PRINT INK a;AT r,c;"■": BEE
P 0.05,10
 190 LET a=a+1
 200 PRINT INK a;AT r,c;"■": BEE
P 0.05,30
 210 NEXT a
 220 NEXT n
 230 GO TO 10



0 OK, 0:1
```

El programa usa POINT para localizar caracteres INK sobre una pantalla PAPER. Las líneas 20 a 50 visualizan minas (cuadrados) azul oscuro sobre una pantalla amarilla. La línea 60 fija la posición de partida del dragaminas (un símbolo "copyright"). Si la posición de partida coincide con alguno de los cuadrados azules aleatorios, se visualiza una nueva serie de cuadrados azules aleatorios. Las líneas 80 a 150 forman una rutina que ya hemos visto varias veces. Hace que el dragaminas rebote en la pantalla de lado a lado y de arriba abajo. Cada vez que el dragaminas alcanza uno de los cuatro bordes de la pantalla, cambia su dirección y suena un BEEP. Este tiene un tono más alto en los choques laterales que en los verticales.

Para cada posición en que se instala el símbolo del dragaminas, la línea 120 convierte sus coordenadas de fila y columna en coordenadas x,y gráficas del punto medio de la posición. La línea 130 después comprueba si ésta es color PAPER o INK. Si es PAPER, el programa retrocede a la línea 80, borra el dragaminas y lo revisualiza en una nueva posición (calculada por la línea 110). Si es INK, la línea 130 restituye el valor 1 y el programa salta a la subrutina de explosión en la línea 190. Esta visualiza dos caracteres gráficos en una gama de colores diferentes, acompañado de efectos de sonido. Después de cada explosión, el programa retrocede a la línea 10 y comienza de nuevo:

PANTALLAS DEL DRAGAMINAS



El número de minas se fija en 5 de manera que el dragaminas tenga suficiente espacio para atravesar la pantalla varias veces antes de que choque con algo. Vd. puede provocar una explosión aumentando este número en la línea 20. El "enmascaramiento" de la pausa en la línea 140 también acelerará el programa.

Usar POINT en rutinas como esta le permite programar el ordenador para tomar una decisión compleja con una sencilla serie de mandatos. POINT es muy útil en juegos en los que se controla un objeto que hay que conducir por la pantalla y evitar paredes u otros obstáculos. Puede probar su práctica usando POINT para dirigir el programa a un fracaso o a una subrutina de penalización cuando Vd. entre dentro de INK.

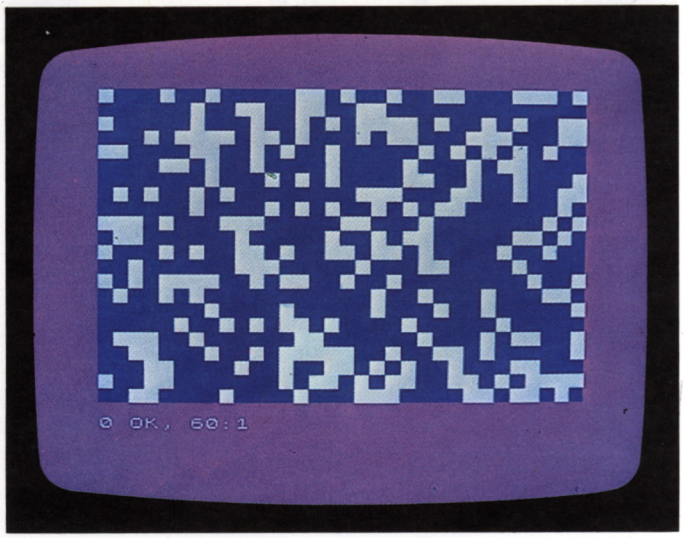
MODELOS CON SIMETRIA

Si con el Spectrum produce un modelo aleatorio, la pantalla será impredecible por completo. Con una técnica sencilla, es posible generar una pantalla con resultado simétrico. Una pantalla impredecible de un modelo aleatorio en toda la pantalla, lo que es útil para producir un fondo de efecto de "galaxia" para un juego de guerra de las estrellas. Pero con una pantalla simétrica, la primera parte del programa produce un modelo aleatorio en un cuarto de pantalla; los otros tres cuartos repiten este modelo en cada una de las tres esquinas de la pantalla con efecto similar a la reflexión en un espejo.

Para ver cómo hacer esto, primero necesita un programa que produzca una pantalla aleatoria, lo que es muy fácil en el Spectrum:

PROGRAMA DE PANTALLA DE AZAR

```
  10 BORDER 3: PAPER 1: INK 7: C
LS
  20 FOR n=1 TO 300
  30 LET c=INT (RND*32)
  40 LET r=INT (RND*22)
  50 PRINT AT r,c;"■"
  60 NEXT n

0 OK, 0:1
```



Este corto programa produce repetidas coordenadas de fila y columna para una posición aleatoria en la pantalla. Aunque la columna más a la derecha está numerada como 31, la línea 30 incluye el número de columna 32 ya que la función INT redondea los números al entero inferior más próximo y por ello el mayor valor posible de INT(RND*32) es 31. El símbolo de gráficos en la tecla 8 se usa para producir la pantalla. Hay un total de 704 posiciones de caracteres; así como máximo se cubre la mitad de la pantalla. No hay nada que impida que las líneas 30 y 40 produzcan el mismo par de coordenadas en varias ocasiones, por lo que con frecuencia los cuadrados INK ocuparán un área menor que ésta.

Cómo reflejar un modelo de azar

Ahora puede usar el símbolo de gráficos, pero de forma diferente, para convertir este modelo aleatorio en uno de simetría:

PROGRAMA DE PANTALLA SIMÉTRICA

```
  10 BORDER 1: PAPER 0: INK 4: C
LS
  20 FOR n=1 TO 75
  30 LET c=INT (RND*16)
  40 LET r=INT (RND*11)
  50 PRINT AT r,c;"■";AT r,31-c;
"■";AT 21-r,c;"■";AT 21-r,31-c;"
■"
  60 NEXT n

0 OK, 0:1
```

Este programa divide la pantalla en cuatro cuartos iguales. Todas las coordenadas producidas por las líneas 30 y 40 caen en el cuarto superior izquierdo; se copian en las posiciones correspondientes de los otros tres cuartos. Como cada par de coordenadas produce la visualización de cuatro símbolos en la pantalla, el programa sólo necesita realizar la cuarta parte de bucles del programa anterior. El valor máximo de n en la línea 20 se reduce por tanto a 75, pero —como antes— se obtienen 300 imágenes. La primera pantalla siguiente muestra un proceso típico del programa; la segunda, el resultado de adaptar el programa cambiando la línea 10 así:

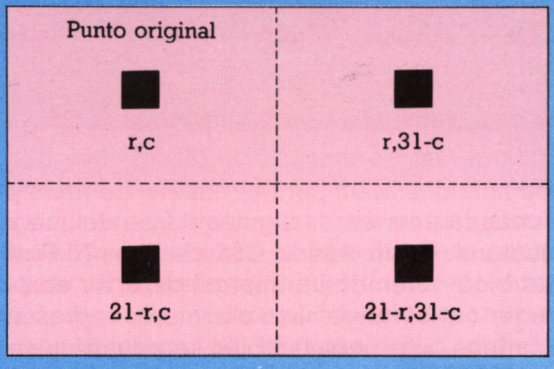
```
10 BORDER 1:PAPER 0:INK 6:CLS
```

y después usando un carácter de gráficos más pequeño (tecla 3). El programa producirá una pantalla diferente cada vez que se procese:

PANTALLAS SIMÉTRICAS

El programa resuelve las posiciones simétricas de cada imagen original ejecutando algunas sencillas operaciones aritméticas sobre las coordenadas r,c. La primera imagen de espejo está en la misma fila que la original pero en el lado opuesto de la pantalla y sus coordenadas son, pues, r,31–c. Cada punto se refleja en otros dos en la mitad inferior de la pantalla, distando del fondo lo mismo que los superiores del tope; así pues, sus coordenadas son: 21–r,c y 21–r, 31–c, en la línea 50.

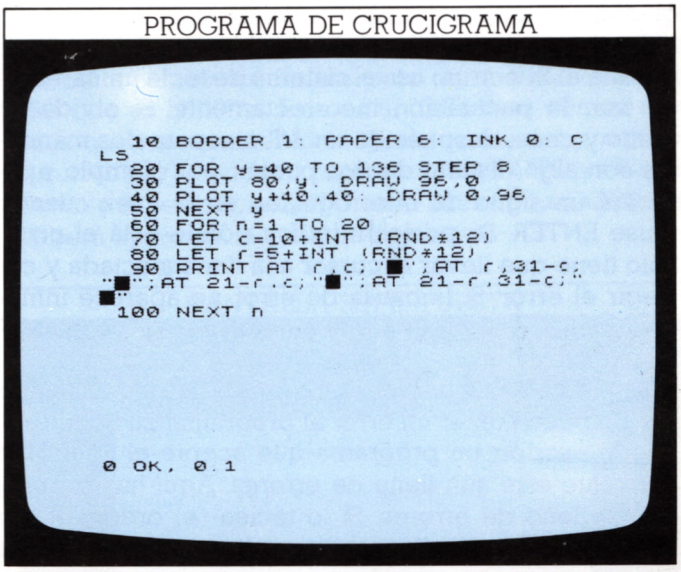
USO DE COORDENADAS SIMÉTRICAS

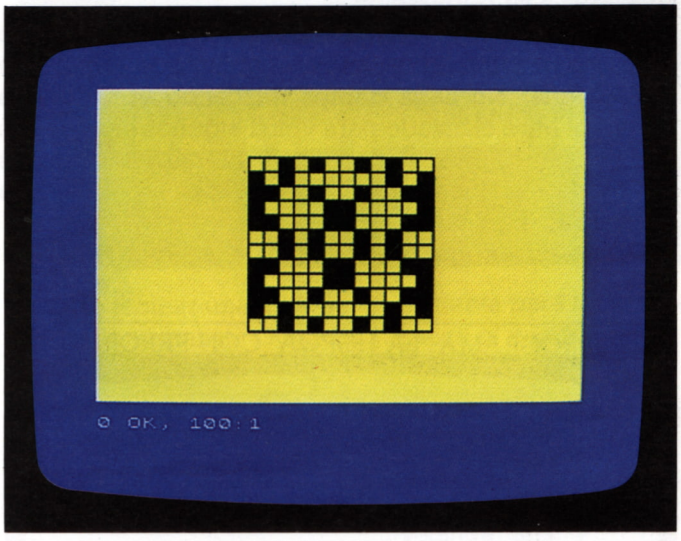


Cómo programar una red de palabras cruzadas

La mayoría de los crucigramas se construyen con redes simétricas, y reflejando los cuadrados negros en una esquina del crucigrama, puede programar que el ordenador complete la visualización. De nuevo se posicionan los cuadros originales al azar:

PROGRAMA DE CRUCIGRAMA





Las líneas 20 a 50 dibujan la red de 12×12 cuadrados. Se elige este espaciado de líneas para que cada cuadro blanco que cae dentro de la rejilla tenga la misma altura y anchura que un carácter de teclado (8×8 pixels gráficos). Las sentencias RND de las líneas 70 y 80 se eligen de manera que las coordenadas de azar producidas por ellas caigan siempre en la red de crucigrama. La línea 90 visualiza el punto original y sus tres imágenes de espejo, como en el programa anterior. Donde se visualice un cuadrado gráfico, este ocupa un cuadro blanco en la red.

BUSQUEDA DE ERRORES

El Spectrum es mucho más útil que muchos microordenadores en el rastreado de errores o "fallos" en los programas. En algunos micros se puede teclear casi todo y sólo se descubre que el programa no tiene sentido al procesarlo y tener un listado de todos los errores cometidos. El Spectrum comprueba línea a línea los errores antes de que pueda introducirlos en un programa.

Aunque no pueda escribir mal una palabra clave, porque el Spectrum usa el sistema de tecla única, es fácil usar la puntuación incorrectamente. Si olvida un punto y coma después de un AT, o separa dos mandatos con algo distinto de dos puntos, por ejemplo, aparecerá un signo de interrogación en la línea cuando pulse ENTER. Su posición indica dónde está el error. Sólo tiene que llevar el cursor a la línea afectada y corregir el error. Si la causa de error no aparece inmediatamente, compruebe que ha usado bien los mandatos en cuestión.

Este sistema de comprobar entradas no significa que Vd. no pueda tener un error al programar su Spectrum. Puede escribir un programa que acepte el Spectrum pero que esté aún lleno de errores. Aquí hay un programa lleno de errores. Si lo teclea (el ordenador le aceptará todas las líneas) después puede ver cómo proceder a su depuración.

El programa es el juego «La horca» de la pág. 27, pero escrito e introducido precipitadamente, de forma que no funciona. ¡No haga trampa buscando el programa correcto! Mire el listado para ver si algunos errores son claros; cuántos puede encontrar y intente ver cómo los corregiría. Después compruebe sus resultados frente a los errores explicados aquí.

Teclee el programa como está e intente usarlo:

PROGRAMA "LA HORCA" CON ERRORES

```
  10 BORDER 0: CLS : PRINT AT 1,
12;"HANGMAN"
  20 PRINT AT 10,2;"Ask a friend
 to type a word"
  30 PRINT AT 12,3;"or phrase fo
r you to guess"
  40 PRINT AT 18,4;"DON'T LOOK A
T THE SCREEN"
  50 PRINT AT 20,0;"Press ENTER
when you're finished"
  60 INPUT a$
  70 LET l=LEN a$
  80 FOR n=1 TO l
  90 IF a$(n)=" " THEN PRINT AT
11,(32-l)/2+n;"■"
 100 PRINT AT 11,(32-l)+n;"j": N
EXT l
 110 PRINT AT 2,12;"HANGMAN";AT
5,6;"-:letter    ■:space"
 120 PRINT AT 18,6;"     Try a le
tter     ";AT 19,0;"Press 1 to gu
ess the whole thing"

scroll?
```

```
 130 INPUT t$
 140 IF t$="2" THEN STOP
 150 IF t$="1" THEN GO TO 190
 160 PRINT AT 16,12;"score=";s
 170 FOR n=1 TO l
 180 IF t$=a$(n) THEN PRINT AT 1
1,(32-l)+n;t$
 190 NEXT n
 200 GO TO 130
 210 PRINT AT 18,6;"Try the whol
e thing": INPUT t$
 220 IF t$=a$ THEN PRINT AT 11,1
2;"CORRECT";AT 13,12;"score=";s+
1
 230 GO TO 120



0 OK, 0:1
```

Cuando intente procesar el programa verá que sube el cuadro de título y Vd. es invitado a introducir la cadena de prueba. Pero cuando pulsa ENTER, aparece el informe de error "1 NEXT without FOR, 100:2". Significa que el programa paró en la 2.ª sentencia de la línea 100. LISTe el programa; verá que un bucle que contiene n comienza en la línea 80, pero la sentencia NEXT en línea 100 dice NEXT l, debiendo decir NEXT n.
Corríjalo y pruebe de nuevo:

PANTALLA DE PROGRAMA "FRACASADO"

```
                HANGMAN


  Ask a friend to type a word
                          jjjjjjj
    or phrase for you to guess


     DON'T LOOK AT THE SCREEN
Press ENTER when you're finished
```

Esta vez el cuadro de título y entrada de cadena de prueba funciona bien, pero el cuadro de título permanece cuando empieza la siguiente fase del juego. Esto se soluciona rápido. Añada :CLS a la línea 70. Puede haber también obtenido un informe de error en pantalla "B Integer out of range". Los caracteres representantes de la cadena de prueba también se visualizan en la posición errónea. Pueden haber salido del lateral de la

pantalla como se muestra en la última pantalla de la página anterior.

La línea 100 debe visualizar (PRINT) todos los caracteres en la mitad de la pantalla. Debería ser:

```
(32-1)/2+n
```

pero en el programa se olvidó dividir por 2.

Las instrucciones de pantalla le dicen a Vd. que un guión representa una letra, pero en su lugar se ha visualizado una fila de j. Además, si ha introducido una cadena de prueba que contiene espacios, habrá visto que no aparecen los símbolos gráficos de cuadros negros.

La línea j apareció porque el programador olvidó pulsar la tecla SYMBOL SHIFT para el guión. Corríjala y vea de nuevo la línea 90. Esta comprueba si existe un espacio en la cadena de prueba y, si es necesario, visualizar un cuadro negro, pero es enmascarado por cualquier impresión de la línea 100. Si Vd. añade :NEXT n al final de la línea 90, cuando se visualiza un cuadro negro, el programa se moverá al siguiente valor de n y al siguiente carácter en la cadena de prueba.

Otros procesos de prueba

Ahora, cuando procesa el programa, tan pronto como introduce su primer adivinanza de una letra, obtiene un informe de error "2 Variable not found, 160:1". La única variable en la línea 160 es s, que representa la puntuación. Parece sencillo —visualiza la puntuación. Esta línea, en efecto, es correcta. El error consiste en que el ordenador no tiene un valor inicial para s, que pueda visualizar. Así, añada :LET s=0 al final de la línea 70. Cuando haya tecleado estas correcciones, procese de nuevo el programa:

PANTALLA PROGRAMA "FRACASADO"

Ahora, cuando teclee una letra adivinada correcta, probablemente aparecerá a la derecha de la pantalla. Por otro lado, si representa una letra que aparece al final de la cadena de prueba, puede aún tener un informe de error "B Integer out of range, 180:2", debido a que el ordenador ha intentado visualizar el carácter fuera de la pantalla. De nuevo se ha omitido la división por 2 de la expresión de columna, que debería ser:

```
(32-1)/2+n
```

El juego ahora parece funcionar, pero si mira la puntuación, verá que no aumenta. Debe aumentar en 1 después de cada respuesta, por lo que debe añadir LET s=s+1: al comienzo de la línea 160.

Aunque ahora las respuestas de una letra funcionan adecuadamente, el ordenador rehúsa responder cuando pulsa 1 para adivinar toda la cadena. La línea 150 debe hacer que el programa salte a la rutina de adivinar de la palabra completa, pero en realidad llega a la 190, que es NEXT n. Este es un fallo muy común después que ha renumerado algunas líneas del programa y quizás haya olvidado renumerar los GOTO y GOSUB. Cambie la sentencia GOTO en la línea 150 a GOTO 210. Deberá aparecer la pantalla siguiente:

PANTALLA SUPERPUESTA

Intente teclear una adivinanza correcta para toda la cadena. Se visualiza "CORRECT" sobre el cuadro anterior del juego. Esto se arregla cambiando la línea 220 a:

```
220 IF t$=a$ THEN CLS: PRINT AT 11,12;
    "CORRECT"; AT 13,12;"score=";s+1
```

Ahora, el programa rehúsa parar. Cuando Vd. ha hecho una respuesta correcta el programa recomienza inmediatamente. Así, añada :STOP al final de la línea 220. El programa deberá ahora marchar adecuadamente.

Si está desarrollando un programa, la comprobación constante debe casi evitar que se deslicen errores en el listado final. Cuando esté probando un programa que ha escrito póngase en todas las situaciones que puede encontrar en su uso. Si supone que tiene la solución para evitar el "fracaso" en algunas circunstancias, pruébelas, desde luego. Probando el final de un programa con GOTO asegurará que cada rutina funcione realmente antes de que Vd. pase a la siguiente.

PROGRAMAS DE VELOCIDAD MEJORADA

Una característica del BASIC es que permite escribir programas sin mucha planificación previa. Con otros lenguajes se necesita un enfoque mucho más organizado. Aunque el BASIC es de fácil uso no fomenta la mejor programación.

Donde se revela la buena o mala programación es en la velocidad de proceso. Aunque esta no sea de mucha importancia en un programa corto, adquiere significación cuando sus programas resultan más largos y complejos. Vd. querrá procesarlos tan rápidamente como sea posible. Aquí podrá comparar un programa pobremente escrito con uno muy trabajado.

Un programa "lento"

El siguiente programa es del tipo del usado en la página 23 para mostrar cómo las matrices producen hojas de cálculo en la pantalla:

PROGRAMA "LENTO" DE TABLA DE IMPUESTOS

```
  10 REM Tax table
  20 DEF FN t()=(65536*PEEK 2367
4+256*PEEK 23673+PEEK 23672)/50
  30 LET T=FN t()
  40 DIM p(8): DIM n(8)
  50 DATA 1.20,2.30,1.10,2.00,1.
80,2.90,1.45,2.05
  60 DATA 14,16,24,17,15,11,14,2
0
  70 FOR a=1 TO 8
  80 READ p
  90 LET p(a)=p
 100 NEXT a
 110 FOR a=1 TO 8
 120 READ n
 130 LET n(a)=n
 140 NEXT a
 150 CLS
 160 FOR y=20 TO 164 STEP 16
 170 PLOT 28,y: DRAW 200,0
 180 NEXT y

scroll?
```

```
 190 PLOT 28,20: DRAW 0,144: PLO
T 68,20: DRAW 0,144: PLOT 92,20:
 DRAW 0,144
 200 PLOT 140,20: DRAW 0,144: PL
OT 180,20: DRAW 0,144
 210 PLOT 228,20: DRAW 0,144
 220 PRINT AT 2,6;"£";AT 2,9;"No
";AT 2,13;"SUB";AT 2,18;"TAX";AT
 2,23;"TOTAL"
 230 FOR a=1 TO 8
 240 PRINT AT 2*a+2,4;p(a)
 250 PRINT AT 2*a+2,9;n(a)
 260 PRINT AT 2*a+2,12;p(a)*n(a)
 270 PRINT AT 2*a+2,18;INT (15*p
(a)*n(a)+0.5)/100
 280 PRINT AT 2*a+2,23;p(a)*n(a)
+INT (15*p(a)*n(a)+0.5)/100
 290 NEXT a
 300 PRINT AT 21,4;"Ti e taken:
";FN t()-T


0 OK, 0:1
```

El programa toma una lista de precios unitarios, números de partidas y, dado el índice de impuestos vigente, calcula el coste total de las partidas, en forma de cuadro. Para probar la velocidad de proceso del programa, las líneas 20 y 30 empiezan temporizando el programa tan pronto como comienza y la línea 300 indica el tiempo total de proceso.

PANTALLA "LENTA" DE TABLA DE IMPUESTOS

£	No	SUB	TAX	TOTAL
1.2	14	16.8	2.52	19.32
2.3	16	36.8	5.52	42.32
1.1	24	26.4	3.96	30.36
2	17	34	5.1	39.1
1.8	15	27	4.05	31.05
2.9	11	31.9	4.79	36.69
1.45	14	20.3	3.04	23.34
2.05	20	41	6.15	47.15

```
Time taken  2.2600002

0 OK, 300:1
```

Los precios unitarios se almacenan en las sentencias DATA en la línea 50, y los números de cada partida en la 60. Ambas matrices se dimensionan por la línea 40. El bucle en las líneas 70 a 100 "lee" los datos de precio unitario en la matriz p, unidimensional de 8 elementos. Un bucle similar en las líneas 110 a 140 "lee" los datos de números en la matriz n. Las líneas 150 a 220 borran la pantalla, dibujan la red de filas-columnas y visualizan las cabeceras de columnas "No" para el número y "SUB" para subtotal. Las líneas 230 a 290 calculan y visualizan los valores para rellenar la red.

Mirando el listado, puede haberse dado cuenta de que los dos bucles usan la variable (a) sobre la misma gama de valores (1 a 8). Habría sido posible combinar estos bucles y ahorrar espacio y tiempo. También el número de fila dado por 2*a+2 se calcula de nuevo en cada línea. Esto lleva algún tiempo. La misma expresión (p(a)*n(a)) aparece tres veces en las líneas 270 a 290, calculada cada vez que se requiere. Todo ello muestra que el programa se ha pensado deficientemente y necesita mejoras; tarda entre dos segundos y un cuarto en procesarse. Vea ahora cuánto puede ahorrar con una mejor técnica de programación.

Técnicas para ganar tiempo

Aquí está de nuevo el programa, pero de forma modificada. Da la misma pantalla, pero en algunos sitios las sentencias son diferentes:

PROGRAMA MEJORADO DE TABLA DE IMPUESTOS

```
  10 REM Fast Tax table
  20 DEF FN t()=(65536*PEEK 2367
4+256*PEEK 23673+PEEK 23672)/50
  30 LET T=FN t()
  40 DIM p(8): DIM n(8)
  50 DATA 1.20,14,2.30,16,1.10,2
4,2.00,17,1.80,15,2.90,11,1.45,1
4,2.05,20
  60 FOR a=1 TO 8
  70 READ p,n
  80 LET p(a)=p: LET n(a)=n
  90 NEXT a
 100 CLS
 110 FOR y=20 TO 164 STEP 16
 120 PLOT 28,y: DRAW 200,0
 130 NEXT y
 140 PLOT 28,20: DRAW 0,144: PLO
T 68,20: DRAW 0,144: PLOT 92,20:
 DRAW 0,144: PLOT 140,20: DRAW 0
,144: PLOT 180,20: DRAW 0,144: P
LOT 228,20: DRAW 0,144

scroll?
```

```
 150 PRINT AT 2,6;"£";AT 2,9;"No
";AT 2,13;"SUB";AT 2,18;"TAX";AT
 2,23;"TOTAL"
 160 FOR a=1 TO 8
 170 LET r=2*a+2: LET b=p(a)*n(a
): LET d=INT (15*b+0.5)/100: LET
 e=b+d
 180 PRINT AT r,4;p(a);AT r,9;n(
a);AT r,12;b;AT r,18;d;AT r,23;e
 190 NEXT a
 200 PRINT AT 21,4;"Time taken:
";FN t()-T



0 OK, 0:1
```

Lo primero que notará es que se han quitado diez líneas del listado. Para conseguirlo, se han quitado cálculos innecesarios y reunido sentencias similares en una sola línea.

Hasta la línea 50 los dos listados son idénticos; todos los datos se han escrito y reorganizado en esta línea, lo que es necesario al reescribir los bucles que cargan los datos en las dos matrices. Hay un bucle sencillo en líneas 60 a 90, ahorrando cuatro líneas. Ambos, precio y número, son "leídos" por READ p,n en línea 70; por ello los datos en la línea 50 deben presentarse así —un precio seguido por un número, etcétera.

Las líneas 110 a 150 son idénticas a las 160 a 220 del primero, salvo que varias sentencias PLOT y DRAW se han escrito en una sola eliminando tres líneas. El ordenador puede manejar sentencias múltiples en una sola línea más rápidamente que varias separadas en líneas diferentes.

Ahora hay un mayor cambio en cómo se rellena la red con valores. Al empezar cada bucle, una vez establecido el valor de a por la línea 160, se hacen todos los cálculos necesarios para ese bucle una vez y para todos en la línea 170. Las variables usadas funcionan de la siguiente forma:

VARIABLES DE LA TABLA DE IMPUESTOS

El programa usa cuatro variables para fijar posiciones y llevar los números calculados para su uso en el cuadro

Variable(s)	Función
r	Fija el número de fila en que se ha de visualizar el dato.
b	Almacena el producto de precio unitario y número de partida (columna SUB).
d	Almacena importe del impuesto al 15 por 100.
e	Almacena coste total (subtotal+impuesto)

La línea 180 es ahora mucho más sencilla, al haberse hecho ya todos los cálculos. Sólo hay que visualizar la variable correcta en su posición adecuada en la red. Si procesa el programa en su forma mejorada, la pantalla mostrará el tiempo así ganado:

PANTALLA MEJORADA DE TABLA DE IMPUESTOS

£	No	SUB	TAX	TOTAL
1.2	14	16.8	2.52	19.32
2.3	16	36.8	5.52	42.32
1.1	24	26.4	3.96	30.36
2	17	34	5.1	39.1
1.8	15	27	4.05	31.05
2.9	11	31.9	4.79	36.69
1.45	14	20.3	3.05	23.35
2.05	20	41	6.15	47.15

```
Time taken:  1.9799995
0 OK, 200:1
```

Los cambios han eliminado más de un cuarto de segundo en este corto programa, más del 12 por 100. Puede imaginar el ahorro en un programa más largo con muchos más datos numéricos o con gran cantidad de cálculos.

Al escribir un programa más largo esté pendiente de las repeticiones en las líneas. Con alguna planificación puede ahorrar tiempo de proceso y memoria. Cualquier rutina grande que se repita convendrá convertirla en una subrutina con GOSUB, y los cálculos que aparezcan mucho, realizarse con FN. Con todo ello y varios bucles se acelera un programa. Como la velocidad de proceso de sus programas se aminora por la conversión en código-máquina, los ahorros en BASIC siempre tienen importancia.

TRUCOS E IDEAS

Cómo guardar (SAVE) pantallas

A veces Vd. puede escribir un programa que produce una pantalla que más adelante le gustaría volver a ver. Con el Spectrum no necesita procesar el programa de nuevo para hacerlo. Los mandatos directos siguientes:

```
SAVE "display" SCREEN$
```

pueden usarse con un magnetófono para almacenar la pantalla llamada "display" o con cualquier otro nombre en una cinta. El ordenador transmitirá a la cinta la información sobre cada "pixel" en la pantalla. Cuando quiera verla de nuevo, pásela como siempre pero con estos mandatos:

```
LOAD "picture" SCREEN$
```

El ordenador explorará gradualmente a través de la pantalla de televisión, visualizando los "pixels" como en la pantalla original. Con este mandato, Vd. puede producir gráficos sin usar líneas de programación, y luego almacenarlos en cinta de la misma forma que un programa.

Problemas con funciones

Dos funciones, SQR y ↑, incorporadas en el Spectrum, pueden a veces originar problemas en sus programas. Como el cuadrado de un número es siempre positivo, no puede usarse SQR con un número negativo, por eso si Vd. está trabajando con SQR, asegúrese de que no ocurra. El problema con la función exponente es menos claro; podría intentar producir un número +1 o -1 (puede usarse en un programa para producir líneas o caracteres sobre la pantalla de forma impredecible): La función exponente debe producir un número positivo o un número negativo, +1 o -1, cada vez, pero esto no sucede. No se debe a que falle la matemática sino a que, igual que con SQR, la función exponente del Spectrum no trabaja con un número negativo. El resultado siempre será positivo —lo cual no es de gran ayuda para los movimientos arriba o abajo. El mejor medio de producir el efecto es establecer un valor decimal de 0 a 0.99999999 con RND, y después usar una línea IF... THEN para producir +1 o -1, como estos:

```
200 LET a=RND
210 IF a>0.5 THEN LET a=1:IF a<=0.5 THEN
    LET a=-1
```

Usada SQR con un número negativo producirá un informe de error; no así la función exponente por lo que no puede darse cuenta al principio por qué un programa está procesándose de forma extraña.

GRÁFICOS ALEATORIOS CON EXPONENTES

```
10 PLOT 128,88
20 LET x=(RND*88)*-1↑(RND*2)
30 LET y=(RND*88)*-1↑(RND*2)
40 DRAW x,y
50 GO TO 10



0 OK, 0:1
```

Cómo establecer límites con movimiento

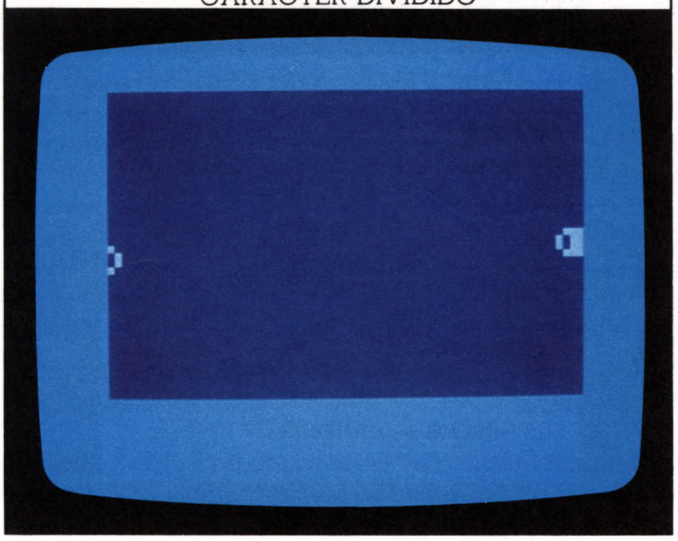
Cuando Vd. inicia gráficos de movimiento, puede encontrar que no se comportan como desea cuando alcanzan el borde de la pantalla. Aquí hay un programa que muestra este problema.

El programa anima un pequeño símbolo de platillo volante construido por dos filas de tres caracteres cada una con un espacio a cada lado; éstos borran las imágenes anteriores cuando la nueva imagen avanza un espacio a la izquierda o derecha de la última posición. La línea 40 fija la posición inicial a (cambio de posición del platillo) a 1. La línea 80 fija la posición de la siguiente visualización. La línea 100 comprueba si el platillo ha alcanzado el borde de la pantalla (c=0 o c=31) y si lo fue, cambia la dirección del platillo. Después retrocede a la línea 50 para visualizar el nuevo:

PROGRAMA DE MOVIMIENTO

```
 10 BORDER 5: PAPER 1: INK 7: C
LS
 20 LET a$="   ":
 30 LET b$="   ":
 40 LET r=10: LET c=0: LET a=1
 50 PRINT AT r,c;a$
 60 PRINT AT r+1,c;b$
 70 BEEP 0.02,5
 80 LET c=c+a
 90 BEEP 0.02,7
100 IF c=31 OR c=0 THEN LET a=-
a
110 BEEP 0.02,9
120 GO TO 50



0 OK, 0:1
```

CARÁCTER DIVIDIDO



Pero, como puede ver, no funciona. El platillo parece enrollarse bajando a la fila siguiente y después retirarse hacia atrás. Es un problema corriente con programas de animación.

La línea 100 se ocupa de cómo se comporta el platillo en los bordes de la pantalla. Recuerde que c representa el número de columna del primer, o más a la izquierda, carácter del platillo. Cuando el último carácter alcanza el borde derecho de la pantalla, c vale 27, no 31. Para rectificar esto cambie la línea 100:

```
100 IF c=27 OR c=0 THEN LET a=-a
```

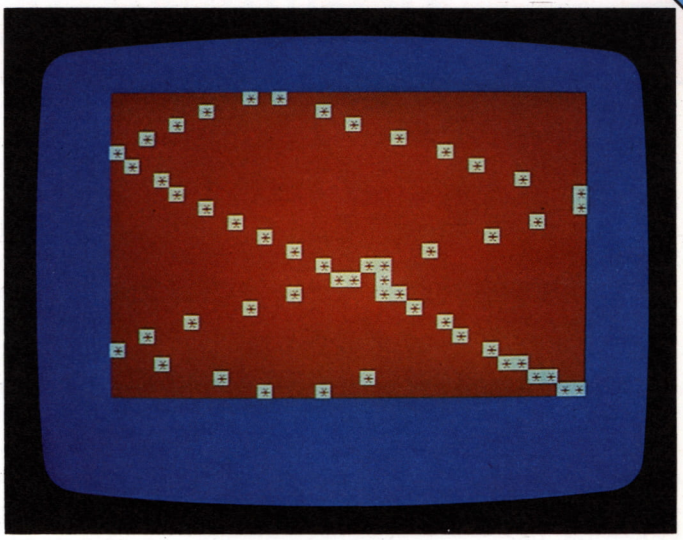
El programa ahora toma en cuenta el tamaño del objeto que se animará.

Cómo determinar el borde de la pantalla

El siguiente programa es similar al anterior, que movía un objeto por la pantalla, pero ahora el objeto consta de un solo carácter que se mueve en ambas direcciones horizontal y vertical:

PROGRAMA COMPROBADOR DE BORDE

```
  10 BORDER 1: PAPER 2: INK 7: C
LS
  20 LET r=11: LET c=16: LET v=1
: LET h=1
  30 PRINT AT r,c;" "
  40 IF INKEY$="v" AND v>0 THEN
LET v=v+0.1
  50 IF INKEY$="h" AND h>0 THEN
LET h=h+0.1
  60 LET r=r+v: LET c=c+h
  70 IF c=0 THEN LET c=0: LET h=
-h
  80 IF c=31 THEN LET c=31: LET
h=-h
  90 IF r=0 THEN LET r=0: LET v=
-v
 100 IF r=21 THEN LET r=21: LET
v=-v
 110 PRINT AT r,c;"■"
 120 BEEP 0.02,60-5*r
 130 GO TO 30

0 OK, 0:1
```



Cada vez que se borra el carácter por la línea 30, la 60 cambia los valores de fila y columna por v y h respectivamente. Estos son al principio 1. Debido a esto y al efecto de las líneas 70 y 80, el carácter rebota en la pantalla como lo haría un balón en el interior de una caja vacía.

Las líneas 40 y 50 le permiten cambiar la velocidad del balón. Si pulsa la tecla V y el balón va hacia abajo, v se incrementa en 0.1, acelerando el balón. Análogamente, si pulsa la tecla H y el balón va hacia la derecha, se aumenta su velocidad horizontal. pero pruébelo. Cuando pulsa V o H, el balón se dispara hacia el borde de la pantalla, y el programa emite un mensaje de error —uno de los dos: "B Integer out of range, 110:1" o "5 Out of screen, 110:1". Si Vd. no toca el teclado, el programa marcha correctamente, probando que las líneas de animación se escribieron bien, pero por alguna razón no trabajan con INKEY$.

Considere por un momento qué sucedería si el balón estuviese en la posición 20,10 yendo hacia abajo a la derecha. En el mismo instante Vd. pulsa V. La línea 40 aumenta v de 1 a 1.1. La línea 60 cambia r a 21.1 y c a 11. La línea 100 comprueba si r=21; no lo es. Así, la línea 110 intenta visualizar el balón en 21.1.11, produciendo el mensaje de error "Out of screen" (Fuera de pantalla).

El problema es que Vd. ya no puede predecir el valor exacto de r o c cuando el balón alcanza un borde. Así, cambie algunas líneas para enfrentarse a la situación y condiciones que no conoce —es decir, ni r ni c pueden tener valores menores que 0, r no puede ser mayor que 21, ni c mayor que 31. Las líneas ahora son:

```
 70 IF c<0 THEN LET c=0:LET h=-h
 80 IF c>31 THEN LET c= 31:LET h=-h
 90 IF r<0 THEN LET r=0:LET v=-v
100 IF r>21 THEN LET r=21:LET v=-v
```

Una vez que haya hecho estos cambios, el programa se comportará correctamente en los bordes.

CONVERSION DE PROGRAMAS

Uno de los problemas del Basic es que existe en muchas formas diferentes o "dialectos". El Basic del Spectrum incluye algunos mandatos —BORDER, PAPER, INK, por ejemplo— que no tienen otros micros. Otros pueden tener efectos que se pueden conseguir con el Spectrum, pero con técnicas totalmente diferentes; no hay sitio más evidente que en programas gráficos. Esto dificulta la "movilidad" de los programas —es decir, usar los mismos programas en diferentes ordenadores. Muchos programas de *software* comercial sólo valen para un tipo de máquina.

Vd. puede considerar conveniente adaptar un programa de una tienda o de un amigo utilizado en máquinas diferentes para que pueda procesarse en su Spectrum. Para hacer esta "conversión", no sólo necesita saber en qué difieren las máquinas sino también comprender por qué y cómo trabaja un programa. Después puede dividir el programa en bloques o subrutinas y finalmente considerar línea a línea. Pero, de la misma manera que no se suele hacer una traducción palabra a palabra, tampoco puede "traducir" cada sentencia de programa "ajeno" directamente a la sentencia equivalente de su ordenador. Puede ser más económico en tiempo y más eficaz reescribir una sección del programa usando los mejores mandatos disponibles en su máquina.

Puntos a vigilar para la conversión

Uno de los aspectos más variables del BASIC es la puntuación y el espaciado. El sistema de comprobación de errores del Spectrum le asegurará que no introduce ninguna línea con defectos de escritura o puntuación, pero Vd. no querrá experimentar hasta tener todo correcto. Punto, dos puntos y punto y coma se usan de forma diferente en muchas máquinas, y necesitará frecuentes cambios de puntuación antes de poder usar las líneas en el Spectrum.

Recuerde que es probable que cualquier coordenada de texto o gráfico necesite cambiarse. Como los diversos ordenadores tienen diferentes resoluciones de pantalla, las coordenadas usadas para producir las pantallas rara vez podrán emplearse sin cambios en los programas de conversión; para ello le ayudará tener una nota de los límites de red de texto y gráficos de los otros micros.

Al convertir programas hay que estar al acecho de los mandatos ligados al sistema operativo de la máquina que no tendrán sentido para el Spectrum; y, recíprocamente, mandatos como PEEK y POKE del Spectrum no se pueden usar en un programa convertido para otras máquinas, ya que se refieren a las direcciones de memoria del Spectrum. Tenga cuidado con los mandatos que utilizan un reloj de ordenador: necesitará reescribir completamente las rutinas para usar el sistema temporizador del Spectrum.

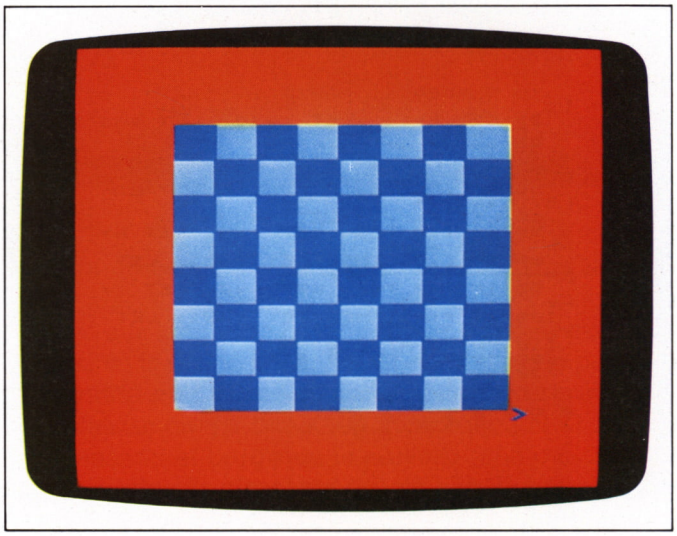
La lista de estos problemas es extensa, pero una vez que haya intentado la conversión de algún programa, aprenderá pronto qué es BASIC estándar y qué es "dialecto". Es bueno, al convertir programas, ver el manual del otro ordenador para descubrir los mandatos que parezcan desconocidos; conocerlos de antemano le facilitará el trabajo.

Listados convertidos y originales

El listado siguiente es un ejemplo de un programa escrito para otro ordenador, el Micro Acorn BBC. Establece un tablero de juegos —rutina que podría querer usar en su Spectrum, pero que no consigue funcionar en él en esa forma:

PROGRAMA TABLERO DE JUEGOS DEL MICRO BBC

```
>LIST
  10 MODE2:VDU5
  20 PROCSCREEN
  30 GCOL0,4:Y=832
  40 FOR ROW=1 TO 7 STEP 2
  50 LEFT=240:RIGHT=840
  60 PROCBOARD
  70 LEFT=340:RIGHT=940:Y=Y-80
  80 PROCBOARD
  90 Y=Y-80
 100 NEXT ROW
 110 END
 120 DEF PROCSCREEN
 130 GCOL0,129:CLG:GCOL0,6
 140 MOVE 240,832:MOVE 1040,832
 150 PLOT 85,240,192:PLOT 85,1040,192
 160 ENDPROC
 170 DEF PROCBOARD
 180 FOR X=LEFT TO RIGHT STEP 200
 190 MOVE X,Y:MOVE X+100,Y
 200 PLOT 85,X,Y-80
 210 PLOT 85,X+100,Y-80
 220 NEXT X
 230 ENDPROC
>_
```

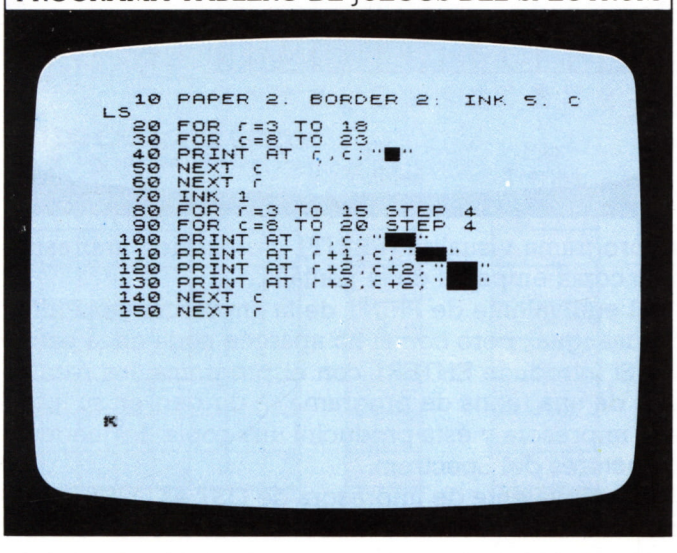


Casi todo en este programa es peculiar del BBC y no puede usarse en el Spectrum. Los mandatos de color y la forma en que llama a los procedimientos (clases especiales de subrutinas) son completamente diferentes. Por ello es necesario comprender cómo trabaja el programa antes de proceder a convertirlo. Le ayudará ver el programa marchando más que trabajar sólo con el listado del mismo.

El modo más sencillo de convertir este programa es tomar cada bloque, identificar su función y escribir una rutina que la efectúe en el BASIC Spectrum. Las líneas 10 a 110 forman el programa principal, que salta a dos subrutinas llamadas PROCSCREEN y PROCBOARD y que son llamadas por su nombre en vez del número de línea. La primera dibuja un gran cuadrado azul claro sobre fondo rojo. El programa principal dibuja a lo largo de la pantalla una fila de cuadrados coloreados alternados (líneas 50 y 60), después la mueve abajo (línea 70) y dibuja una segunda fila, que está desplazada a la derecha de la primera. Con cuatro pares de estas filas se hace un tablero de ajedrez. El BBC construye cada cuadrado con un par de triángulos, algo no disponible en el Spectrum.

Ahora puede reescribir el programa para procesarlo en el Spectrum. Como la gran parte del programa está en BASIC BBC, no se gana nada intentando traducirlo línea a línea. Es más rápido y fácil examinar el programa y después encontrar cómo podría el Spectrum producir mejor el mismo resultado:

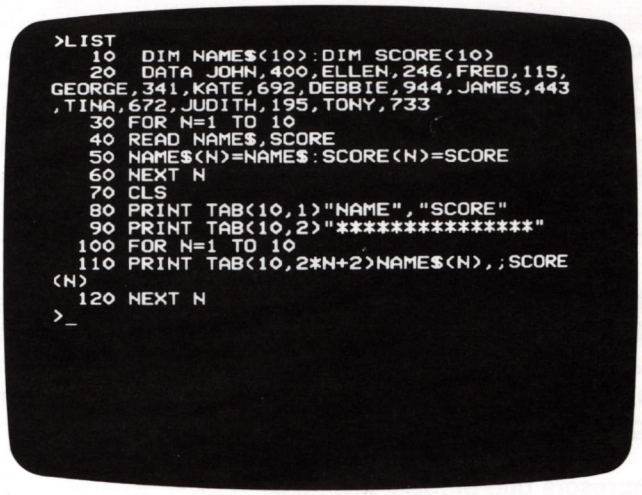
PROGRAMA TABLERO DE JUEGOS DEL SPECTRUM



Más diferencias del BASIC

El siguiente programa, también escrito en BASIC BBC, lista diez nombres en la pantalla. Cada uno va acompañado de un número —podría ser una lista de las mejores puntuaciones de un juego. A primera vista el método parece muy familiar ya que usa matrices, y debería ser adaptable para el Spectrum:

PROGRAMA DE MATRICES DEL MICRO BBC



El Spectrum puede producir esto pero esté al acecho de un típico problema de ocultación. El Spectrum dimensiona las matrices de forma diferente. El BBC dimensiona la matriz de nombre así:

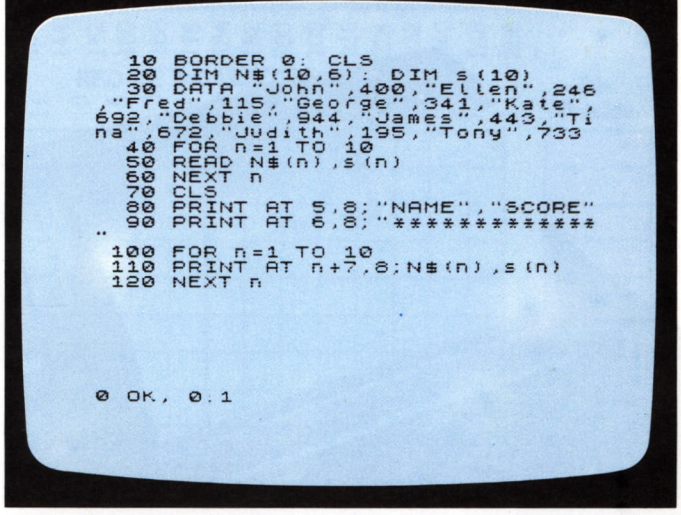
```
10 DIM NAME$(10)
```

pero el Spectrum también requiere la longitud de la cadena, así:

```
20 DIM N$(10,6)
```

También las cadenas de los DATA en el Spectrum deben encerrarse entre comillas. Por lo demás, los programas son muy similares:

PROGRAMA DE MATRICES DEL SPECTRUM



Cuando intente convertir programas que parecen usar mandatos compatibles con el Spectrum, cuide de no caer en la trampa de suponer que operan igual. Palabras como DIM, INKEY$, TAB, etc., pueden hacer la conversión difícil si no se da cuenta de que muchos ordenadores usan los mismos mandatos de forma diferente.

USO DE IMPRESORA

Aunque se puede escribir, editar, procesar programas en pantalla y almacenarlos en magnetófono, una copia en papel es el mejor medio de examinar atentamente un listado. También le permite conservar el producto de un proceso de programa, lo que es muy útil si se quieren volver a ver los resultados. Así, tarde o temprano querrá añadir una impresora a su sistema.

La impresora Sinclair es de tipo termoelectrostático. Chispas que saltan desde un par de agujas móviles metálicas sobre papel especial cubierto de aluminio, queman el aluminio en puntos que revelan debajo una capa de tinta. La forma de las chispas, y el carácter consiguiente, se controlan por el ordenador.

Impresora que permite mandatos

La impresora se conecta al ordenador con un conector en el panel posterior del Spectrum. Cuando está enchufada no imprime nada, a menos que Vd. se lo diga. Las sentencias PRINT envían los caracteres a la pantalla de televisión, ignorándolas la impresora completamente.

Hay varios medios de remitir información a la impresora. El programa de la derecha permite verlos funcionar, ofreciéndole una elección de tres mandatos de impresora. Estos le permiten usar la impresora selectivamente. Puede producir una versión impresa de lo que el programa visualiza (PRINT), o un listado impreso, o una copia en papel de la pantalla.

El equivalente de PRINT de la impresora es LPRINT. Se usa igual, pero con él no aparece nada en la pantalla. Si introduce ENTER1 con el programa, los resultados de una rutina de programa se dirigen, en su lugar, a la impresora y ésta producirá una copia del juego de caracteres del Spectrum.

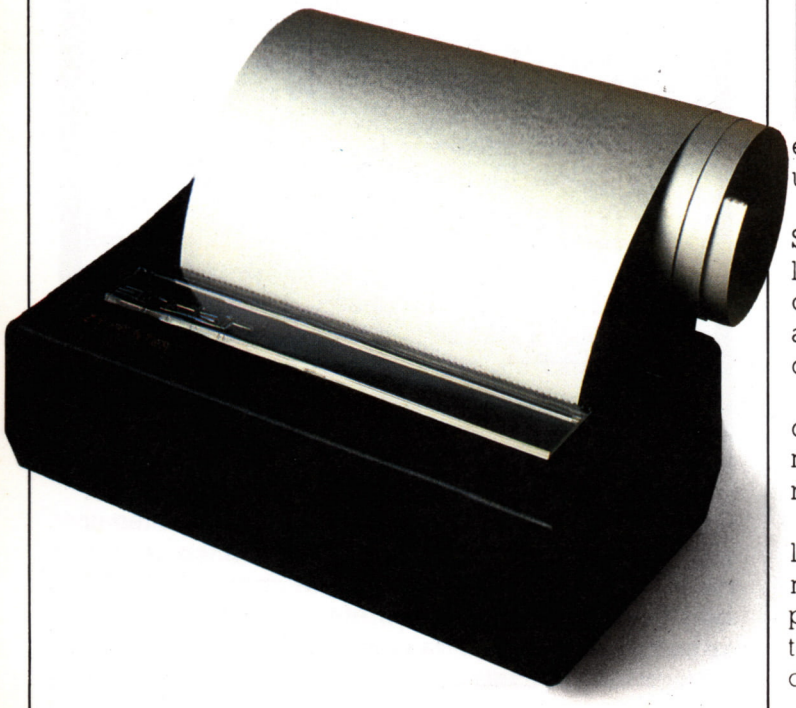
El equivalente de impresora de LIST es LLIST, seleccionado en el programa introduciendo 2. Otra vez, este mandato sólo trabaja sobre la impresora y no producirá un listado en la pantalla.

El método final del uso de la impresora está controlado por el mandato COPY, seleccionado en el programa introduciendo 3. Esto hace que aparezca en la impresión cualquier cosa que se visualice sobre la pantalla. Este mandato se utiliza para copiar gráficos, cuadros, etc.

PROGRAMA DE IMPRESIÓN

```
 10 PRINT AT 4,8;"Enter Choice"
 20 PRINT AT 6,8;"1  To LPRINT"
 30 PRINT AT 8,8;"2  To LLIST"
 40 PRINT AT 10,8;"3  To COPY"
 50 INPUT a
 60 GO TO 100*a
100 LPRINT "This will print the
Spectrum character set";
110 FOR f=32 TO 255
120 LPRINT CHR$ f;
130 NEXT f
140 STOP
200 LLIST
210 STOP
300 CLS
310 PLOT 93,15
320 DRAW 160,88
330 DRAW -176,0
340 DRAW 160,-88
350 DRAW -72,136
360 DRAW -72,-136
370 COPY

0 OK, 0:1
```

```
Enter Choice
1  To LPRINT
2  To LLIST
3  To COPY

3
```

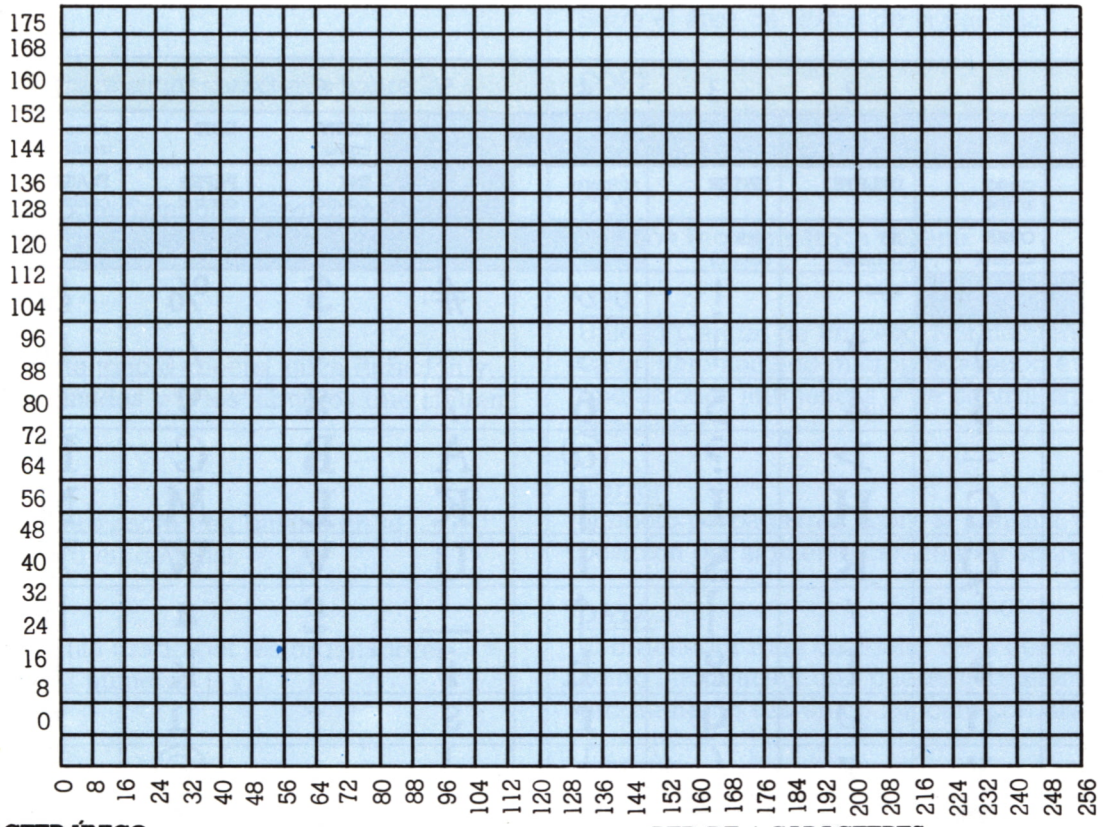


La impresora ZX La impresora usada con el Spectrum se controla con 3 palabras claves que pueden utilizarse en programas.

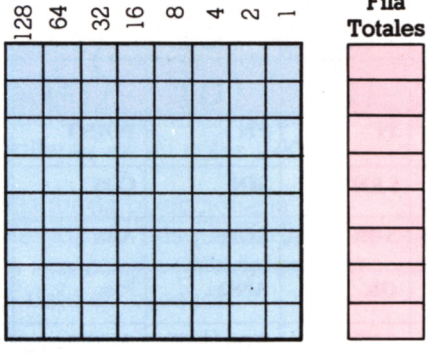
REDES DE GRAFICOS Y DE CARACTERES

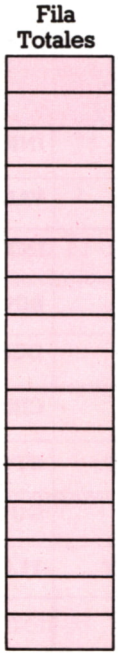
La red de abajo muestra las coordenadas de la pantalla cuando se usan los mandatos de gráficos. Un punto sobre la pantalla se define por dos coordenadas x,y. La primera, abscisa, fija la posición horizontal referida al lado izquierdo de la pantalla. Un carácter visualizado en la pantalla ocupa un área que tiene una anchura y altura de 8 unidades gráficas. No puede imprimirse en las dos líneas inferiores de la pantalla.

RED DE COORDENADAS GRÁFICAS

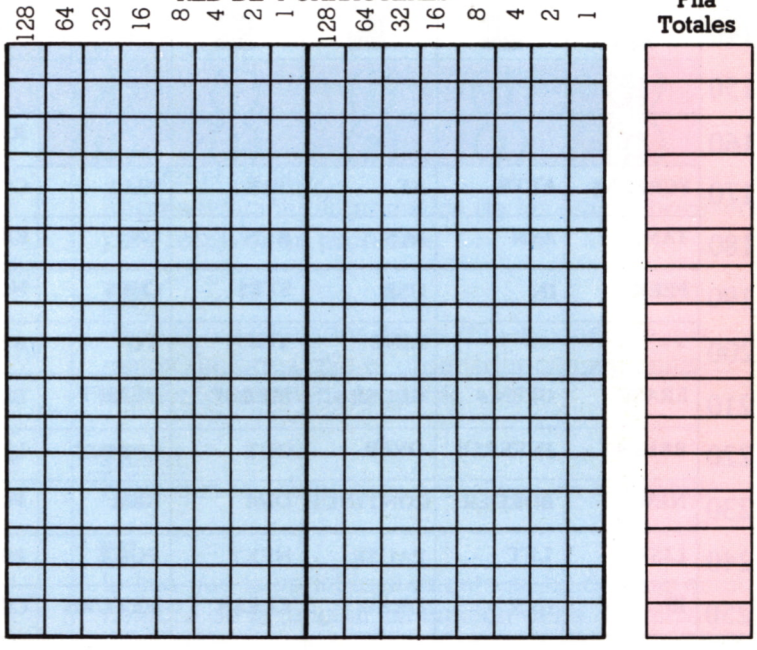


RED DE CARÁCTER ÚNICO





RED DE 4-CARACTERES



Rejillas de caracteres. Estas rejillas pueden usarse para diseñar o bien un único carácter *(encima)* o un símbolo hecho hasta con 4 caracteres *(derecha)*. Vd. puede rellenar su diseño en las rejillas y usar las columnas azules para listar los totales de fila. Estos se usan en los programas con los mandatos POKE USR. Las teclas A a U están libres para programar caracteres definidos por el usuario.

EL JUEGO DE CARACTERES SPECTRUM

Cada símbolo o palabra clave que usa el Spectrum está representado por un número de código. Hay 256 números de código en total (0-255), cada uno de los cuales puede convertirse en un simple octeto (byte) de ocho dígitos binarios. La letra S, por ejemplo, se especifica por CHR$ 83, o CHR$ BIN 1010011. El ordenador reconoce la forma binaria de números de código como instrucciones o información necesaria para realizar programas. Los códigos 33 a 126 se asignan a los caracteres especificados por las normas ASCII. Las palabras clave y símbolos gráficos del Spectrum se especifican por códigos "de repuesto".

	0	1	2	3	4	5	6	7	8	9
0							PRINT coma	EDIT	cursor izquierdo	cursor derecho
10	cursor abajo	cursor arriba	DELETE	ENTER	número		INK control	PAPER control	FLASH control	BRIGHT control
20	INVERSE control	OVER control	AT control	TAB control						
30			space	!	"	#	$	%	&	'
40	(	)	*	+	,	-	.	/	0	1
50	2	3	4	5	6	7	8	9	:	;
60	<	=	>	?	@	A	B	C	D	E
70	F	G	H	I	J	K	L	M	N	O
80	P	Q	R	S	T	U	V	W	X	Y
90	Z	[	\	]	↑	_	£	a	b	c
100	d	e	f	g	h	i	j	k	l	m
110	n	o	p	q	r	s	t	u	v	w
120	x	y	z	{	|	}	~	©		
130										
140					(a)	(b)	(c)	(d)	(e)	(f)
150	(g)	(h)	(i)	(j)	(k)	(l)	(m)	(n)	(o)	(p)
160	(q)	(r)	(s)	(t)	(u)	RND	INKEY$	PI	FN	POINT
170	SCREEN$	ATTR	AT	TAB	VAL$	CODE	VAL	LEN	SIN	COS
180	TAN	ASN	ACS	ATN	LN	EXP	INT	SQR	SGN	ABS
190	PEEK	IN	USR	STR$	CHR$	NOT	BIN	OR	AND	<=
200	>=	<>	LINE	THEN	TO	STEP	DEF FN	CAT	FORMAT	MOVE
210	ERASE	OPEN #	CLOSE #	MERGE	VERIFY	BEEP	CIRCLE	INK	PAPER	FLASH
220	BRIGHT	INVERSE	OVER	OUT	LPRINT	LLIST	STOP	READ	DATA	RESTORE
230	NEW	BORDER	CONTINUE	DIM	REM	FOR	GO TO	GO SUB	INPUT	LOAD
240	LIST	LET	PAUSE	NEXT	POKE	PRINT	PLOT	RUN	SAVE	RANDOMIZE
250	IF	CLS	DRAW	CLEAR	RETURN	COPY				

GLOSARIO

Las entradas en **negrita** son palabras clave del BASIC.

AND
Permite que un programa tome un curso de acción particular sólo si se cumplen dos condiciones. Es una extensión de **IF... THEN.**

AT
Se usa con **PRINT** para situar caracteres sobre la pantalla.

BASIC
Beginners' All-purpose Symbolic Instruction Code; es el lenguaje de programación de alto nivel más utilizado.

BEEP
Hace sonar un corto tono o «beep», cuya duración y tono están determinados por los números que siguen al mandato.

BIN
Convierte un número escrito en binario en el equivalente número en decimal.

Binario
Un sistema de cuenta usado por los ordenadores basado sólo en dos números, 0 y 1.

Bit
Dígito binario —0 ó 1.

BORDER
Cambia el color del área del borde de la pantalla.

BRIGHT
Vuelve caracteres especificados a un tono más brillante de su color INK.

Búsqueda biseccionada
Método de búsqueda de un dato específico. El dato se bisecciona (divide en dos) hasta que se encuentra el dato en cuestión.

Byte (u octeto)
Grupo de ocho bits.

Cadena
Secuencia de caracteres tratados como elemento simple.

Chip
Una pieza que contiene un circuito electrónico completo. También se llama circuito integrado (IC).

CHR$
Traduce el número que sigue de un código de carácter al carácter equivalente.

CIRCLE
DIbuja un círculo del tamaño especificado con su centro en un punto especificado de la pantalla.

CLS
Borra el área de textos de la pantalla.

COPY
Instruye a una impresora para imprimir una copia de la pantalla.

COS
Función trigonométrica coseno.

CPU
Unidad Central de Proceso. Normalmente contenida en un *chip* llamado microprocesador; éste efectúa las operaciones aritméticas y de control en el resto del ordenador.

Cursor
Símbolo parpadeante sobre la pantalla, que indica la posición del siguiente carácter en aparecer.

DATA
El ordenador trata cualquier cosa que sigue a **DATA** como información que puede necesitar después en el programa. Se usa en conjunción con **READ**.

DEF FN
Define una función usada en cualquier parte de un programa por el mandato **FN**.

Depuración
Proceso de eliminar los errores o fallos de un programa.

Diagrama de flujo
Representación diagramática de las etapas necesarias para resolver un problema.

DIM
Informa al ordenador de las dimensiones de una matriz de forma que el ordenador conoce cuántos elementos contiene la matriz.

FLASH
Hace que los caracteres centelleen en la pantalla.

FN
Indica que la variable siguiente se ha de usar como nombre de la función. La función debe definirse por una sentencia **DEF FN**.

FOR... NEXT
Bucle que repite una secuencia de sentencias de programa un número especificado de veces.

GOSUB
Hace que el programa salte a una subrutina que empieza en el número de línea que sigue al mandato. La subrutina debe siempre terminarse por **RETURN.**

GOTO
Hace que un programa salte al número de línea siguiente al mandato.

Gráfico de barras
Tipo de gráfico en que los datos numéricos se representan por columnas.

Gráfico de sectores
Pantalla gráfica de datos numéricos en forma de un círculo dividido. El tamaño de cada sector refleja el tamaño del número que representa.

Hardware
Maquinaria física de un sistema de ordenador, distinto de los programas que le hacen realizar el trabajo.

IF... THEN
Incita al ordenador a tomar un curso de acción particular si se detecta la condición especificada.

INK
Cambia el color del texto y gráficos que aparecen en la pantalla.

INKEY$
Vigila el teclado para ver si se ha pulsado cualquier tecla y, en caso afirmativo, devolver el carácter.

INPUT
Instruye al ordenador para que espere algún dato, procedente del teclado, que se ha de usar después en el programa.

INT
Convierte un número con decimales en el número total.

Interface
La conexión de *hardware* y *software* entre un ordenador y otra pieza de equipo.

INVERSE
Conmuta el color **PAPER** al color **INK** y viceversa.

K
Abreviatura de kilobyte (1024 bytes).

LEN
Cuenta el número total de caracteres en una cadena que le sigue.

LET
Asigna un valor a una variable.

LIST
Hace que el ordenador visualice el programa en curso en su memoria.

LLIST
Instruye a una impresora para que imprima un listado del programa en curso en la memoria.

LOAD
Transfiere un programa desde un magnetófono a la memoria del ordenador.

LPRINT
Envía lo que siga al mandato a una impresora, en vez de a la pantalla, de forma que se produzca sólo una copia de papel.

Matriz
Colección de datos organizados de forma que cada elemento quede etiquetado y pueda manejarse por separado.

MERGE
Permite que se cargue un segundo programa en el ordenador desde un magnetófono sin borrar el programa en curso en la memoria, en tanto que no coincidan los números de línea.

NEW
Elimina un programa de la memoria del ordenador.

OR
Permite que un programa tome un curso de acción particular si se cumple una de las dos condiciones especificadas. Es una extensión de **IF... THEN.**

OVER
Permite visualizar nuevos caracteres encima de caracteres existentes.

PAPER
Cambia el color del fondo de la pantalla.

PAUSE
Detiene un programa durante un período fijado por un número medido en quincuagésimos de segundo.

PEEK
Informa del número almacenado en una posición especificada en la memoria.

PLOT
Hace que aparezca un punto en la pantalla en el punto especificado.

POINT
Informa si el punto en las coordenadas que siguen se visualiza como color **INK** o como color **PAPER.**

POKE USR
Almacena un número que reprograma una tecla para producir un carácter definido por el usuario. Por su cuenta, **POKE** pone un número en una posición especificada de la memoria.

PRINT
Hace que lo que le sigue aparezca sobre la pantalla.

RAM
Random Acces Memory (memoria de acceso aleatorio). Una memoria cuyo contenido se borra cuando se corta la corriente. (Ver ROM).

RANDOMIZE
Fija el punto en el cual empezará la secuencia aleatoria **(RND).**

READ
Instruye al ordenador para que tome la información de una sentencia **DATA.**

REM
Permite al programador añadir notas o comentarios a un programa. El ordenador ignora lo que quiera que siga al mandato.

RESTORE
Reajusta el punto a partir del cual los elementos DATA se han de leer (READ), de forma que los elementos puedan usarse más de una vez en un programa.

RETURN
Termina una subrutina (Ver también **GOSUB).**

RND
Produce números entre el 0 y el 1, al azar.

ROM
Read Only Memory (Memoria de Sólo Lectura). Una memoria que se programa permanentemente por el fabricante y cuyo contenido sólo puede leerse por el ordenador del usuario.

SAVE
Registra sobre una cinta un programa en curso en la memoria del ordenador. El programa se identifica por un nombre de archivo.

Sentencia
Una instrucción en un programa. Puede haber más de una sentencia en cada línea del programa.

SCREEN$
Mantiene los detalles de una pantalla visualizada de forma que se puedan guardar (SAVE) o cargar (LOAD).

SIN
Función trigonométrica seno.

Software
Programas del ordenador

SQR
Produce la raíz cuadrada del número que sigue.

STEP
Fija el tamaño de escalonamiento en un bucle **FOR... NEXT.**

STOP
Para un programa y visualiza el número de línea en que aparece.

Subrutina
Parte de un programa que puede llamarse cuando sea necesario, para producir una pantalla particular o realizar repetidamente un número de cálculos, por ejemplo.

Sintaxis
Reglas que gobiernan la manera en que deben ponerse las sentencias junto con el lenguaje del ordenador.

TAB
Se usa con **PRINT** para especificar a qué distancia en la línea han de aparecer caracteres. Sirve para disponer columnas.

VAL
Valora una cadena de números, y produce un número.

Variable
Una zona etiquetada en la memoria del ordenador, en la que puede almacenarse información y recuperarse después en un programa.

VERIFY
Comprueba que un programa que está en curso en memoria ha sido registrado correctamente en un magnetófono usando **SAVE.**

INDICE

Las entradas principales aparecen en **negrita**

Curso Visual en Pantalla

"...para todo el que quiera llegar a ser mejor programador. El método de enseñanza es infalible"

Nigel Searle, Director
SINCLAIR RESEARCH

Original y apasionante curso de programación para los usuarios del ZX Spectrum.

Más de 150 fotografías de pantallas con listados de programas que le muestran exactamente lo que aparecerá en su pantalla.

Otros títulos de la serie:

- ZX Spectrum +,
- Apple IIe,
- Commodore 64

MicroTextos